V. CHETVERIKOV

DATA PROCESSING FOR CONTROL AND MANAGEMENT

MIR PUBLISHER / MOSCOW

V. CHETVERIKOV

DATA PROCESSING FOR CONTROL AND MANAGEMENT

MIR PUBLISHERS / MOSCOW

The book is conserned with automated management information and control systems. A good deal of space is devoted to codes and coding, data communication, and data preparation for entry into a computer. Separate chapters deal with punched-card and punched-tape devices, I/O operations and peripherals. Special emphasis is placed on A/D and D/A conversion and converters.

The book will be of value to students majoring in management information and control systems, and also to practical engineers in this field.

ИЗДАТЕЛЬСТВО МИР

В. Н. ЧЕТВЕРИКОВ

ПРЕОБРАЗОВАНИЕ И ПЕРЕДАЧА ИНФОРМАЦИИ В АСУ

МОСКВА «ВЫСШАЯ ШКОЛА»

V. CHETVERIKOV

DATA PROCESSING FOR CONTROL AND MANAGEMENT

Translated from the Russian
by
B. Kuznetsov

MIR PUBLISHERS · MOSCOW

First published 1977

Revised from the 1974 Russian edition

The Greek Alphabet

Αα	Alpha	Ιι	Iota	Ρρ	Rho
Ββ	Beta	Κκ	Kappa	Σσ	Sigma
Γγ	Gumma	Λλ	Lambda	Ττ	Tau
Δδ	Delta	Μμ	Mu	Υυ	Upsilon
Εε	Epsilon	Νν	Nu	Φφ	Phi
Ζζ	Zeta	Ξξ	Xi	Χχ	Chi
Ηη	Eta	Οο	Omicron	Ψψ	Psi
Θθϑ	Theta	Ππ	Pi	Ωω	Omega

The Russian Alphabet and Transliteration

Аа	a	Кк	k	Хх	kh
Бб	b	Лл	l	Цц	ts
Вв	v	Мм	m	Чч	ch
Гг	g	Нн	n	Шш	sh
Дд	d	Оо	o	Щщ	shch
Ее	e	Пп	p	Ъъ	"
Ёё	e	Рр	r	Ыы	y
Жж	zh	Сс	s	Ьь	'
Зз	z	Тт	t	Ээ	e
Ии	i	Уу	u	Юю	yu
Йй	y	Фф	f	Яя	ya

На английском языке

© Translation into English, Mir Publishers, 1977

PREFACE

In the design of management information and control systems, a good deal of attention is devoted to what are called peripherals — facilities external to the central processor (or computer), whose purpose is to gather, reduce and convey data from source to sink. Peripherals go a long way towards making a success two-way communication between man and machine in systems like MIS's. This is why so much emphasis is placed in this text on these devices that link together man and computer. Through these devices, man can collect and enter any source data and programs, retrieve results, and address the central processor.

In the general case, a MIS may utilize data on the progress of a production process from automatic transducers. If the transducers generate analog signals, that is, signals continuously varying with time, the need arises to convert these signals to numbers before data can be entered in the central processor. In turn, in order that the numbers generated by the computer can be utilized to run the process, they should first be converted back to appropriate analog signals. These two aspects are dealt with in the last two chapters.

Where users are remote from the central processor of a MIS, resort is made to data communication over various communication channels. The physical channels are not discussed in the book, but ample attention is given to the information aspects of data communication.

The book is largely based on the lectures the author read at the Baumann Higher Technical School in Moscow.

The author wishes to thank the faculty members of the Computer Engineering Chair at the Moscow Institute of Electronic Engineering and Professor K. A. Sannikov, D. Sc. (Tech), for their review of the manuscript and for their suggestions, and also V. N. Kononykhin, V. A. Galkin, A. E. Osminin and G. I. Revunkov for their assistance in the preparation of some sections.

CONTENTS

Preface 7

Chapter 1. Information and Its Measure 9

1.1. Presentation of Information 9
1.2. Sampling, Quantizing and Coding 12
1.3. Amount of Information 24
1.4. Entropy in the Continuous Case 31
1.5. ε-Entropy of Differentiable Functions 34

Chapter 2. Communication Channels and Systems 41

2.1. Basic Definitions 41
2.2. Modulation and Demodulation 43
2.3. Interference and Noise 53
2.4. Capacity of a Communication Channel 54
2.5. Structure and Organization of Information Networks 58

Chapter 3. Codes and Error Correction 64

3.0. Definitions 64
3.1. Telegraph Codes 64
3.2. Codes Used for Entry of Data to Digital Computers 67
3.3. Error-Detecting and Error-Correcting Codes 72
3.4. Systematic Codes 81
3.5. Cyclic Codes 88
3.6. Recurrent Codes 99

Chapter 4. Decoding of Redundant Digital Signals 103

4.1. Decoding by Sequence and Symbol Estimation 103
4.2. Error Immunity of Decoding by Sequence Estimation 107
4.3. Wagner's Detection Scheme and Decoding on the Basis of the Reliable Symbols 109
4.4. Erasure Decoding 110
4.5. Probabilistic Decoding of Group Codes 112
4.6. Probabilistic-Algebraic Error Detection 117

Chapter 5. Two-Way Data Communication Systems 119

5.1. Data-Feedback Systems 121
5.2. Decision-Feedback Systems Using Non-Redundant Codes 125
5.3. Receiver-Interlock Systems 128
5.4. Address-RQ Systems 132
5.5. Adaptive Feedback Systems 134
5.6. Data-Communication Equipment 138

Chapter 6. Preparation of Data for Entry into the Computer 148

6.1. Punched Cards and Punched Tape 149
6.2. Card and Tape Punches 157
6.3. Card and Tape Readers 172
6.4. Electric Typewriters 175
6.5. Input Preparation Equipment 182
6.6. Automatic Input Preparation 187
6.7. Teleprinters 192
6.8. Data Loggers 198

Chapter 7. Data Transfer Systems and Devices 205

7.1. Data Transfer in First-Generation Computers 205
7.2. Data Transfer in Second-Generation Computers 206
7.3. Data Transfer in Third-Generation Computers 209
7.4. General Principles of Program Interrupts 212
7.5. Logic Structure of Time-Sharing Systems 216
7.6. Multiplexor and Selector Channels 218
7.7. Data Entry from Punched Tape and Cards 221
7.8. Output Card and Tape Punches 231
7.9. Alphanumeric Printout 240
7.10. Terminal Multiplexor 249

Chapter 8. Analog-to-Digital Conversion 256

8.1. A/D Conversion Errors 257
8.2. Incremental Code-Wheel Converters 258
8.3. Total-Value A/D Converters 260
8.4. Matrix Coding 274
8.5. Indirect Conversion 277
8.6. Multispeed A/D Converters 292
8.7. Reading Methods 297
8.8. Optical Gratings 313
8.9. Voltage-to-Time-to-Digital Converters 320
8.10. Selective-Subtraction A/D Converters 329
8.11. Feedback A/D Converters 333
8.12. Cathode-Ray-Tube A/D Converters 335

Chapter 9. Digital-to-Analog Conversion 338

9.1. Weighted-Voltage D/A Converters 338
9.2. Weighted-Resistor D/A Converters 340
9.3. D/A Converters Using a D.C. Amplifier 345
9.4. Current-Summation D/A Converters 349
9.5. Two-Resistance-Value Converters 351
9.6. Serial Binary Number-to-Voltage Conversion 355
9.7. Number-to-Shaft Position Conversion 356

Appendix 365

Bibliography 369

Index 371

CHAPTER 1

INFORMATION AND ITS MEASURE

1.1. PRESENTATION OF INFORMATION

Information may be defined as any data about an event or an object. The concept of information, as it is understood in cybernetics, is akin to that of reflection dealt with in dialectic materialism. The property of reflection is to be found not only in objects but also in processes; in the latter case it manifests itself as a definite relation, or correspondence, between the states of interacting objects. While philosophers are mainly concerned with qualitative differences between forms of reflection, cyberneticists concentrate on a quantitative treatment.

Whether one deals with correspondence between sensation and reality or between the indication of a voltmeter and the voltage across its terminals, the situations are similar because the former object is reflected, or mapped, into the latter, that is, the latter contains some information about the former. On this basis, we may say that *information* is a mapping of a state of one object into a state of another, provided there is a correspondence between their states. States of an object may be mapped into those of several objects (with a varying degree of accuracy).

It is convenient to use the term *message* for the form in which the information is presented to a communication system. Whatever the contents of a message, it is always conveyed through the system in a form (electrical, aural, light, etc.) called a *signal.*

A signal is always formed when a message is to be conveyed from a sender (source) to a recipient (sink) which are, in the general case, separated in space and time. Therefore, a signal may be defined as a means of conveying information in space and time. However this definition of the signal is purely functional and does not describe it as an object of investigation.

From an analysis of any situation involving the use of signals, we may readily conclude that, although signals are always related to a material object, most of the specific qualities of that object are of minor significance. For example, in learning the contents of a printed text it is unimportant what kind of ink or paper has

been used; differences between texts (signals) are above all recognized from differences in letters, that is, states of the objects. Therefore, we may say that signals are not the objects themselves, but their states. A signal is formed by causing the object to change state. To preserve correspondence between a message and a signal, that is, to preserve the possibility of extracting the original message from the signal received, the latter should be formed according to definite rules. Such a transformation is called *coding*, and the related rules are called a *code*.

Leaving out the physical aspects of a signal, we may note that in the transmission of information from source to sink, a number of tasks has to be tackled in order to make information more convenient to process and convey. This point can best be understood from the following example.

Let a source generate messages as code combinations (words) made up of characters (letters and/or numerals) out of an alphabet X having a total of l characters. Each message may be thought of as being a selection a out of an alphabet A. It may be represented by a selection of characters, x, out of the alphabet X. This decomposition of messages into elementary units greatly simplifies the transformation of a message into a signal. Instead of using a long dictionary (an infinite one in the general case) in order to establish correspondence between all selections out of the alphabet A and signals, it will suffice to set up unique correspondence between signals and a limited number x of elementary units out of the alphabet X. Obviously, the number of signals needed to represent such elementary units will be determined by the number l of characters in the alphabet.

Thus, each selection a out of the alphabet A may be represented as a sequence, $a = (x_1, x_2, \ldots, x_n)$, where n is the word length.

To transmit a message a represented by a string of elementary units x_i, signals $z_i(t)$ should be sent into the communication channel consecutively.

In practice, the number of characters, l, in the alphabet X may be fairly great (decimal notation uses ten numerals, Russian alphabet has 32 characters, etc.). That is why resort is made to a further transformation, called coding. By coding, the alphabet X having l characters is replaced by another alphabet, Y, having m characters (where $m < l$). This coding assigns to each elementary unit, or character, x, a certain series of symbols, y.

In the wider sense, coding has to do with the transformation of a message into a signal. In the narrower sense, this is a process

by which discrete messages are mapped into a series of agreed symbols.

The reduction in the number of characters in the alphabet entails a reduction in the number of distinct signals, $z_i(t)$, necessary for their transmission, and also eases the limitation imposed on the duration of each signal. As a result, transmission is simpler to organize.

For information to be reliably conveyed in space and time, the signals used must be immune to spatial and temporal variations. Qualitatively, this immunity is specified in relation to specific conditions under which the signal is used. In terms of stability, all signals may be divided into static and dynamic.

Static signals are those which utilize stable states of physical objects (printed texts, states of flip-flops, states of a register, the position of a mechanical element, etc.).

Dynamic signals are those which utilize the time-varying states of physical objects (variations in the electromagnetic field, variations in electric parameters, etc.).

Dynamic signals are mainly used to convey information, and static signals, to store it. This division of functions is not mandatory, however, and dynamic signals may well be used to store information and static signals, to convey it.

As to the structure, signals may be divided into continuous and discrete in terms of function and argument. A signal is *continuous* if all the values it can take form a continuum (examples are continuously varying electric current, voltage, or mechanical displacement). A signal is *discrete* if it is limited to a finite (countable) set of values. Signals may further be classed according to the behaviour of the function and its argument.

A continuous function of a continuous argument. At arbitrary instants of time, the function describing such a signal may take on any value out of an infinite set of values within a finite interval

$$F(t)_{\min} \leqslant F(t) \leqslant F(t)_{\max}$$

No limitations are imposed on the behaviour of the function.

Examples are signals generated by pressure, voltage or position transducers as continually varying voltages or currents.

A continuous function of a discrete argument. The function describing such a signal may take on any value out of a continuous set at predetermined instants of time, $t_k = k\Delta t$, where $k = 0, 1, \ldots, n$.

A discrete function of a continuous argument. At any instant of time, the function describing such a signal may take on any value, $F(t)_j$, out of a finite set. Examples are all scale mechanisms from which data at any instant are read off as numerical values accurate to within the least scale division, ΔF.

A discrete function of a discrete argument. The function describing such a signal may take on any value out of a finite set at predetermined discrete instant of time, $F(\Delta t_k)_j$. Such signals are generated by digital computers or clock-controlled digital devices.

Data processing systems usually employ both continuous and discrete signals and have, therefore, to resort to conversions from continuous to discrete form and/or back.

1.2. SAMPLING, QUANTIZING AND CODING

In most cases, information about a physical process is generated by appropriate transducers as signals (variables) which are continuous functions of time, $F(t)$. Before such a continuous (or analog) signal can be presented to a discrete (or digital) communication system or a digital computer, the continuous waveform should be converted to a discrete or digital waveform. This can be done in analog-to-digital converters by techniques known as *sampling* (or time quantization) and *quantizing* (or amplitude quantization). It is possible to apply sampling and quantizing separately and together.

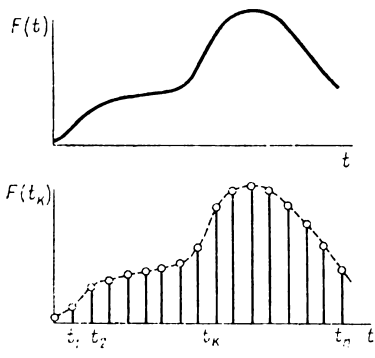

Fig. 1 1. Sampling of a signal

Sampling. In this case, a continuous function, $F(t)$, having an infinite number of values, is replaced by a countable number of instantaneous values, or samples, taken at predetermined intervals (equally or unequally spaced), Δt (Fig. 1.1). Sampling is analogous to amplitude modulation applied to pulses of an infinitesimal duration.

The sampling rate, $f = 1/\Delta t$, is selected so as to facilitate the subsequent reconstruction of the original continuous signal from its samples taken at instants t_k with sufficient accuracy. Generally, an arbitrary continuous function, $F(t)$, can be faithfully reproduced within a finite time interval T, only if samples of this function

are taken at all points of this interval, that is, if there is an infinite number of samples spaced an infinitesimal interval apart.

Reconstruction of a continuous function from a finite number of samples within a finite time interval T produces an error determined by both the number of samples taken within that interval (or the sampling rate) and the method of interpolation adopted. There is, however, a class of functions which can faithfully be reconstructed from a finite number of samples. This is true, for example, of bandlimited functions.

The sampling rate for bandlimited functions can be found on the basis of V. A. Kotelnikov's sampling theorem * which states: *If a continuous time function $F(t)$ contains no frequencies higher than f_s, it is completely determined by giving its instantaneous values, $F(k\Delta t)$, at a series of instants spaced $\Delta t = 1/(2f_s)$ apart.* Here, $k = 0, 1, 2, \ldots, n$ and f_s is the highest frequency present.

By this theorem, any function $F(t)$ extending from 0 to f_s may be expanded into a series such that

$$F(t) = \sum_{-\infty}^{+\infty} F(k\Delta t) \frac{\sin \omega_s (t - k\Delta t)}{\omega_s (t - k\Delta t)} \tag{1.1}$$

where $\omega_s = 2\pi f_s$.

It follows from Eq. (1.1) that any bandlimited function $F(t)$ may be represented by an infinite sum in which each term is expressed by a function of the form

$$z = y (\sin x)/x$$

where

$$y = F(k\Delta t)$$

and

$$x = \omega_s (t - k\Delta t)$$

(Fig. 1.2), and differs from the other terms in amplitude and time shift (phase). The functions defining $\sin x/x$ at sampling instants, that is, at $t = k\Delta t$, take on maximum values equal to unity. On the other hand, the sum, Eq. (1.1), at each kth instant is defined by only one kth term because all other terms vanish at that instant (Fig. 1.3). Over the interval Δt the reconstructed function is defined by all terms.

* Known as Shannon's sampling theorem in the American literature. — Tr.

Thus, a bandlimited function $F(t)$ can be completely specified by giving a finite number of instantaneous values (samples) taken at equally spaced intervals Δt. On the other hand, if we have the numerical values of a function $F(t_k)$ at all sampling points (spaced Δt apart), we can completely reconstruct the original function by adding together functions of the form $z = y\ (\sin x)/x$.

Kotelnikov's theorem applies to a bandlimited function, $F(t)$, that is, a function unlimited in time. In practice, however, one has to deal with physical processes which are time-limited, that is, they have both a beginning and an end. Therefore, functions applicable to them are likewise time-limited. Time-limited functions cannot be bandlimited (that is, their spectral density is non-zero outside the finite interval), which conflicts with Kotelnikov's theorem. The conflict can be resolved, however, by a reasonable assumption, that is, by defining an *effective bandwidth*, that is a frequency range outside which the spectral density falls below some predetermined value. Then Kotelnikov's theorem stating that a function of duration T can be specified by giving a set of $n = 2f_sT$ samples, will hold. However, this does not imply that

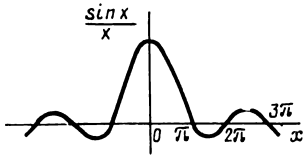

Fig. 1.2. Plot of the function sin x/x

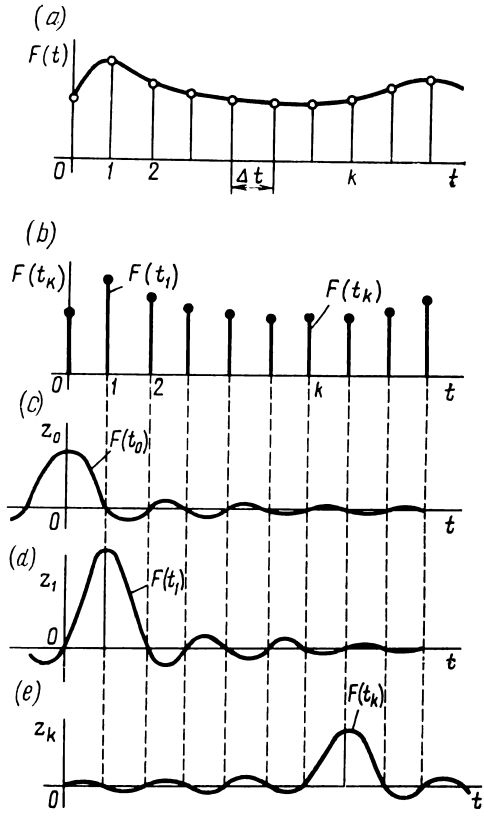

Fig. 1.3. Plots of (*a*) the function $F(t)$; (*b*) of its instantaneous values, $F(t_k)$; and (*c*), (*d*), (*e*) its terms

this function can completely be specified by giving all n samples over the interval T, because the defining functions associated with points outside the chosen interval will likewise affect the reconstructed function (except the sampling points t_k where all defining functions, except one, vanish).

Because the number of terms in the series, Eq. (1.1), is limited to a finite number of values, $F(t_k)$, occurring over the sampling interval T, a sampling error arises, defined as

$$\delta_k = F(t) - \sum_{-m}^{n} D_k \frac{\sin \omega_s [t - k/(2f_s)]}{\omega_s [t - k/(2f_s)]}$$

where $D_k = F(k\Delta t)$.

It may be shown that

$$\delta_k \leqslant \sqrt{2}\, p/\pi \,|\sin \pi f_s t\,| \sqrt{(1/f_s)\frac{T}{T^2 - t^2}} = \varepsilon p \qquad (1.2)$$

where p is the total energy carried by the function $F(t)$, such that

$$p = \int_{-f_s}^{f_s} S(\omega)\, d\omega$$

and

$$\varepsilon = \sqrt{2}/\pi \,|\sin \pi f_s t\,| \sqrt{(1/f_s)\frac{T}{T^2 - t^2}}$$

A plot of ε in Eq. (1.2) is shown in Fig. 1.4. As is seen, when a finite number of terms is used in Kotelnikov's series, the sampling eror is zero only at the sampling points, and it is a maximum between them. The maximum value of the sampling error rises towards the bounds of the interval on which the function is defined.

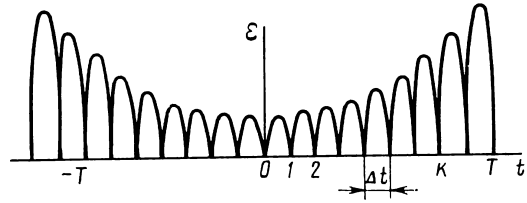

Fig. 1.4. Error plot

So far, the numbers defining a bandlimited function have been its instantaneous values; however, the coefficients of a Fourier series may do as well.

Consider the application of Kotelnikov's theorem to the sampling of an exponential signal such that

$$f(t) = A\exp(-\beta t)$$

or

$$f(t) = A\exp(\beta t)$$

where A is the amplitude and β is $1/\tau$ (where τ is the time constant).

To determine the sampling rate by Kotelnikov's theorem, we limit the bandwidth of the signal to f_s chosen such that the energy of all discarded harmonics will not exceed the energy of the error. Suppose that the error in the signal reconstructed from its samples should not exceed $k\Delta p$, where k is the coefficient and Δp is the quantizing step. Then the total energy of an exponential signal, p_e, may be represented as a sum of two terms

$$p_e = p_s + p_r \tag{1.3}$$

where p_s is the energy of the bandlimited signal, of frequency f_s, and p_r is the energy of the discarded harmonics, equal to the energy of the reconstruction error.

The total energy of the signal, p_e, can be found from its spectral density, $S(\omega)$, which for the exponential functions has the form

$$S(\omega) = A/(\beta + j\omega); \qquad S(\omega) = A/(j\omega - \beta) \tag{1.4}$$

or

$$S(\omega) = \left(A/\sqrt{\beta^2 + \omega^2}\right)\exp[j\arctan(\omega/\beta)] \tag{1.5}$$

For a signal containing all harmonics from 0 to infinity

$$p_e = \int_0^\infty [f(t)]^2\,dt = \frac{1}{2\pi}\int_0^\infty S(\omega)\,S^*(\omega)\,d\omega \tag{1.6}$$

where $S^*(\omega)$ is the complex conjugate of the function $S(\omega)$.

Substituting (1.5) in (1.6) gives

$$p_e = \frac{1}{2\pi}\int_0^\infty \left\{\frac{A}{\sqrt{\beta^2+\omega^2}}\exp[-j\arctan(\omega/\beta)]\,\frac{A}{\sqrt{\beta^2+\omega^2}}\right.$$

$$\left.\times\exp[j\arctan(\omega/\beta)]\right\}d\omega = \frac{1}{2\pi}\int_0^\infty \frac{A^2}{\beta^2+\omega^2}\,d\omega = A^2/4\beta \tag{1.7}$$

Similarly to (1.7), the energy p_s of a bandlimited signal containing no frequencies higher than f_s (such that $2\pi f_s = \omega_s$) is defined as

$$p_s = \frac{1}{2\pi} \int_0^{\omega_s} \frac{A^2}{\beta^2 + \omega^2} d\omega \tag{1.8}$$

Since any value of the signal in the interval between 0 and $n\Delta p$ (where n is the maximum number of quantizing steps on the amplitude scale) is equiprobable, the average value of p_e has the form

$$p_e = (1/n\,\Delta p) \int_0^{n\Delta p} x^2 dx = n^2 \Delta p^2/3 \tag{1.9}$$

The energy of the error can be found by assuming that its any value within the specified limits of $k\Delta p$ is likewise equiprobable. Then the average energy of the reconstruction error will be defined by an equation similar to (1.9):

$$p_r = \frac{1}{k\,\Delta p} \int_0^{k\Delta p} x^2 dx = k^2 \Delta p^2/3 \tag{1.10}$$

Solving (1.9) and (1.10) simultaneously gives

$$p_r = k^2 p_e/n^2 = \delta^2 p_e \tag{1.11}$$

where $\delta = k/n$ is the fractional error.

Substituting the expression for p_e from (1.7) into (1.11) yields

$$p_r = \delta^2 A^2/4\beta$$

Substituting the above values of p_s, p_r and p_e in (1.3) and carrying out appropriate manipulations, we get

$$A^2/4\beta = \frac{1}{2\pi} \int_0^{\omega_s} \frac{A^2}{\beta^2 + \omega^2} d\omega + \delta^2 A^2/4\beta$$

$$(1 - \delta^2)\, A^2/4\beta = \frac{1}{2\pi} \int_0^{\omega_s} \frac{A^2}{\beta^2 + \omega^2} d\omega$$

$$(1 - \delta^2)\, A^2/4\beta = (A^2/2\pi)\,(1/\beta) \arctan\,(\omega_s/\beta)$$

From the last expression, we get

$$\arctan \omega_s/\beta = (\pi/2)(1 - \delta^2)$$

or

$$\omega_s/\beta = \tan \frac{\pi}{2}(1 - \delta^2)$$

Recalling that $\beta = 1/\tau$ and $\omega_s = 2\pi f_s$, we have

$$f_s = (1/\tau 2\pi) \tan \frac{\pi}{2}(1 - \delta^2)$$

By Kotelnikov's theorem, the sampling rate is

$$f = 2f_s = (1/\tau\pi) \tan (\pi/2)(1 - \delta^2)$$

This expression defines the sampling frequency f for a continuous exponential signal at the specified reconstruction error $k\Delta p$ and the maximum range of the parameter $n\Delta p$ or at a specified fractional error δ. Within sampling intervals Δt, the signal should be reconstructed as stated by the theorem, and the quantizing error will be solely due to the fact that the signal is band-limited. Thus, an exponential function with $\tau = 1$ will have the following values of δ and f (see Table 1.1):

Table 1.1

δ %	10	5	1	0.5	0.2	0.1	0.05
f, s^{-1}	20.22	80.9	2021.62	8077.78	50,080.96	195,405.14	705,979.88

As is seen, at $\delta = 0.05\%$, the sampling rate is $f > 700$ kHz, which is a fairly high figure.

Sampling may be looked upon as an approximation in which we seek a function $\varphi(t)$ of class Φ which gives the best approximation (in a specified sense) to the reproduced function $f(t)$ in class F.

The approximating function can be constructed by interpolation, average-power or uniform approximation.

In the case of interpolation, it is sought to construct an interpolation polynomial $\varphi(t)$ of degree n, which takes the same values as the function $f(t)$ at $(n + 1)$ points $t_0, t_1, \ldots, t_n$:

$$\varphi(t_0) = f(t_0), \quad \varphi(t_1) = f(t_1), \ldots, \quad \varphi(t_n) = f(t_n)$$

The problem reduces to finding the coefficients a_i (where $i = 0, 1, \ldots, n$) of the polynomial

$$\varphi(t) = a_0 + a_1 t + a_2 t^2 + \ldots + a_n t^n$$

from the equations

$$\begin{aligned}
a_0 + a_1 t_0 + a_2 t_0^2 + \ldots + a_n t_0^n &= f(t_0) \\
a_0 + a_1 t_1 + a_2 t_1^2 + \ldots + a_n t_1^n &= f(t_1) \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
a_0 + a_1 t_n + a_2 t_n^2 + \ldots + a_n t_n^n &= f(t_n)
\end{aligned}$$

The solution can be found by use of the Lagrange formula for interpolation

$$\varphi(t) = \sum_{i=0}^{n} f(t_i)\,\varphi_i(t)$$

$$= \sum_{i=0}^{n} \varphi(t_i)\,\frac{(t-t_0)(t-t_1)\ldots(t-t_{i-1})(t-t_{i+1})\ldots(t-t_n)}{(t_i-t_0)(t_i-t_1)\ldots(t_i-t_{i-1})(t_i-t_{i+1})\ldots(t_i-t_n)}$$

In the general case, the function $f(t)$ being approximated and the interpolation polynomial $\varphi(t)$ do not coincide at points other than interpolation points t_i ($i = 0, 1, \ldots, n$), and the difference between them, $\delta(t) = f(t) - \varphi(t)$, is the approximation error. This error can be evaluated by the Rolle theorem: if $F(t_k) = 0$ ($k = 0, 1, \ldots, n$) where $t_0 < t_1 < \ldots < t_n$, a point ξ ($t_0 < \xi < t_n$) exists such that $F^{(n)}(\xi) = 0$.

Assuming t to be a fixed number other than interpolation points t_i ($i = 0, 1, \ldots, n$), we introduce a variable T and analyse the function

$$F(T) = f(T) - \varphi(t) - \frac{\delta(T)}{A(t)}\,A(T) \tag{1.12}$$

where $A(T) = (T - t_0)(T - t_1)\ldots(T - t_n)$.

The function $F(t)$ vanishes at the following $(n + 2)$ points:

(a) the interpolation points t_i ($i = 0, 1, \ldots, n$) because $f(t_i) = \varphi(t_i)$ and $A(t_i) = 0$ ($i = 0, 1, \ldots, n$);

(b) at the given point t:

$$F(t) = f(t) - \varphi(t) - \frac{\delta(t)}{A(t)}\,A(t) = 0 \tag{1.12a}$$

By Rolle's theorem, $F^{(n+1)}(T)$ vanishes at some point ξ on the interval $t_0 < T < t_n$.

Differentiating (1.12) gives

$$F^{(n+1)}(T) = f^{(n+1)}(T) - \varphi^{(n+1)}(T) - \frac{(\delta T)}{A(t)} A^{(n+1)}(T)$$

From (1.12a), it follows that

$$A^{(n+1)}(T) = \frac{\delta(t)}{A(t)} (n+1)! \tag{1.13}$$

Since, however, the polynomial $\varphi(t)$ is of degree n, then $\varphi^{(n+1)}(T) = 0$. Therefore,

$$F^{(n+1)}(T) = f^{(n+1)}(T) - (n+1)! \frac{\delta(t)}{A(t)}$$

Then, we may re-write the equality

$$F^{(n+1)}(\xi) = 0$$

as

$$F^{(n+1)}(\xi) = f^{(n+1)}(\xi) - (n+1)! \frac{\delta(t)}{A(t)} = 0$$

Hence,

$$\delta(t) = f^{(n+1)}(\xi) \frac{A(t)}{(n+1)!} \tag{1.14}$$

The upper bound to the error on the interval $t_0 < t < t_n$ is

$$\max |\delta(t)| = \max_{t_0 < t < t_n} |f^{(n+1)}(t)| \left| \frac{A(t)}{(n+1)!} \right|$$

$$= \frac{M}{(n+1)!} \max_{t_0 < t < t_n} \left| \prod_{k=0}^{n} (t - t_k) \right|$$

where $M = \max |f^{(n+1)}(t)|$.

Notably, in the case of linear interpolation ($n = 1$)

$$\delta = \delta(t) \leqslant \frac{\max |f''(t)|}{(1+1)!} \max_{t_0 < t < t_n} |(t - t_0)(t - t_1)|$$

$$= (M/2) \max_{t_0 < t < t_n} |(t - t_0)(t - t_1)|$$

Let us find the maximum value of the term $|A(t)| = |(t - t_0)(t - t_1)|$:

$$dA(t)/dt = t - t_0 + t - t_1 = 2t - t_0 - t_1$$

From the condition $dA(t)/dt = 0$ it follows that $t = (t_0 + t_1)/2$.

Then,

$$\max|A(t)| = \max|(t-t_0)(t-t_1)| = |(t-t_0)(t-t_1)|$$
$$= \left|\frac{t_1-t_0}{2}\,\frac{t_0-t_1}{2}\right| = \frac{1}{4}(t_1-t_0)^2 = \Delta t^2/4$$

Finally, we get

$$\delta \leqslant (M/2)(\Delta t^2/4) = M\,\Delta t^2/8$$

where $M = \max|f^{(n+1)}(t)|$.

If linear interpolation has been chosen and the maximum value of the second derivative $|f''(t)|_{\max}$ of the function is known, the sampling step Δt may be deduced from the following relation:

$$\Delta t = \sqrt{8\delta/|f''(t)|_{\max}} \tag{1.15}$$

It is also assumed that the function $f(t)$ describing the variable being sampled falls in the class of twice differentiable functions and that the maximum value of the second derivative of this function over the sampling interval is known. As calculations show, the sampling rate found by this method within the limits of practical inaccuracy and processes is much lower. For example, the sampling rate for the exponential function $\exp(-t/\tau)$ examined earlier with $\tau = 1$ s is found by the linear interpolation method to be

$$f = 1/\Delta t = \sqrt{|f''(t)|_{\max}/8\delta}$$

and has the values listed in Table 1.2:

Table 1.2

δ, %	10	5	1	0.5	0.2	0.1	0.05
f, s^{-1}	1.12	1.59	3.55	5	8	11.2	15.9

Of a wider interest is the case where the error is a minimum for the specified sampling interval Δt and the known norm $|f''_{\max}(t)|$ defined by the function being interpolated. As follows from (1.14), this can happen when the polynomial $A(t)$ has a minimum value, that is, the polynomial $A(t)$ must have the least norm of all polynomials of degree n between a and b. This property is manifested by Chebyshev's polynomials.

In constructing an interpolating polynomial $\varphi(t)$ coinciding with the function $f(t)$ at points t_i $(i = 0, 1, \ldots, m)$, it is found that the degree n of the polynomial $\varphi(t)$ cannot be smaller than m. With high values of n, the interpolation polynomial becomes extremely unwieldy to develop. Also, good approximation cannot be secured even with a great number of equally spaced points. This is why it will be a good plan to find a polynomial $\varphi(t)$ of degree n lower than m, that will take at points t_i $(i = 0, 1, \ldots, m)$ such values $\varphi(t_i)$ that would depart least of all (in some specified sense) from the values $f(t_i)$ $(i = 0, 1, \ldots, m)$ of the function being interpolated. Most often this problem is solved by the least squares method. By the method of least squares approximation reduces to finding a function $\varphi(t)$ such that the expression $[\varphi(t) - f(t)]^2$ is minimized.

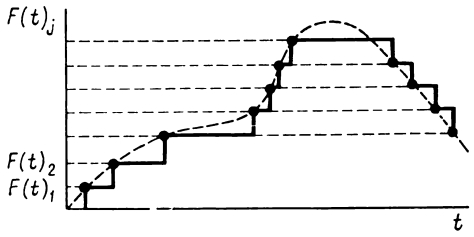

Fig. 1.5. Quantized signal

This leads to a set of simultaneous linear algebraic equations known as the normal equations whose solution does not pose any particular difficulties. This is why the method of least squares has come to be so widely used.

In uniform sampling or approximation, the absolute departure of $\varphi(t)$ from $f(t)$, should be a minimum, that is, $\max|\varphi(t) - f(t)| \leqslant \delta$.

It involves unwieldy equations, because of which its use has been limited.

Quantizing (amplitude quantization). Quantizing consists in that from the continuum of amplitude values of the signal one selects discrete values usually uniformly distributed over the continuum (Fig. 1.5), but their jumps between steps occur at arbitrary times. As a result, a stepped (or rather, staircase) waveform, $F(t)_j$, is derived to represent the continuous function $F(t)$. In contrast to sampling where the samples are basically accurate representations of the continuous function, quantizing yields an approximate representation. The closeness of approximation is dependent on the *quantizing step*, $\Delta F(t)$, defined as the difference between adjacent quantizing levels

$$\Delta F(t) = F(t_k) - F(t_{k-1})$$

The quantizing step is usually chosen from considerations of the requisite accuracy.

A very simple quantizing technique is as follows. An instantaneous value of the function $F(t)$ is compared with a quantizing level (Fig. 1.6). At the instant when the instantaneous value of $F(t)$ becomes equal to that level, it is stored and maintained until the new instantaneous value becomes equal to the next quantizing level.

Thus, if within some time interval the change in the function be less than the quantizing step, the function will be represented by the previous quantized level.

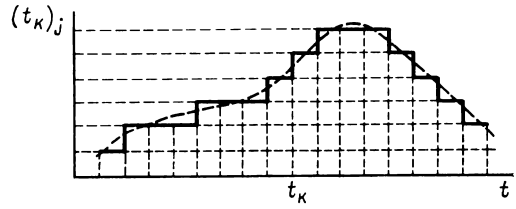

Fig. 1.6. Pulse-code modulation

In another quantizing technique, the instantaneous values of the continuous function are replaced by the nearest quantized levels. The quantizing error within the quantizing step will then be a varying quantity

$$\delta = F(t) - F(t)_k, \qquad \delta \leqslant \Delta F(t) \qquad (k = 0,\ 1,\ \ldots,\ n)$$

This quantizing error is often called the *quantization noise* or distortion. The smaller the quantizing step, the smaller the quantization noise. It may be assumed that quantization noise within a quantizing step or interval has an equiprobable distribution, that is, any value of the function is equiprobable within the quantizing step.

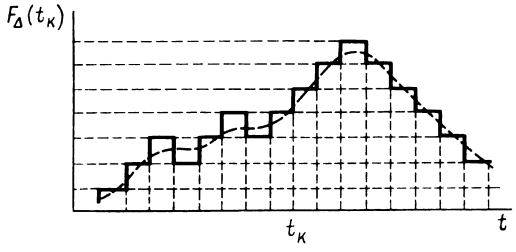

Fig. 1.7. Delta-modulation

Sampling and quantizing. In many cases, sampling and quantizing are used together. Examples are pulse-code modulation and delta modulation.

Pulse-code modulation (*PCM*). In this method, the message waveform is first sampled at a sampling step Δt, the samples are then replaced by the nearest discrete (quantized) levels (see Fig. 1.6).

Delta modulation. In this method, a staircase waveform $F_\Delta(t)$, is constructed by the following rule. At each discrete instant, t_k, the instantaneous value of the continuous function is compared with the previous value of the staircase function, $F_\Delta(t_{k-1})$. If the difference is zero or positive, the staircase waveform is incremented by a quantizing step; if the difference is negative, the staircase waveform is decremented by the same amount (Fig. 1.7).

1.3. AMOUNT OF INFORMATION

In the general case, when information is probabilistic, it is convenient to evaluate it quantitatively in terms of the uncertainty of the stochastic process at a particular stage. This point can best be illustrated by an example.

A person has thought up some number from 1 to N and asked another person to guess it. In his attempt to guess it, the other person asks questions to which the first man gives answers of the "yes" or "no" type. How many questions will the other person have to ask before he guesses the right number?

Obviously, the number of questions will depend on the strategy of guessing. If he chooses to enumerate all numbers, starting with unity, he will have to ask N questions. The number of questions can markedly be cut down by using dichotomy. In brief, it runs as follows. All numbers are divided into two groups, and any one is then submitted for identification. Whatever the answer ("yes" or "no"), only one group will remain. This remaining group is again divided into two, and the procedure is repeated, until the desired number remains alone.

If the total count of numbers N is even, the necessary and sufficient number of questions will be

$$H = \log_2 N$$

if it is odd, then

$$H = [\log_2 N] + 1$$

This is a quantitative estimate of the search.

To grasp the part played here by information, let us approach the same problem from a different angle. Suppose that a source of information generates words consisting of characters from some agreed alphabet and that there is a telegraph transmitter which can only send two digits, a 0 and a 1. So that we can use this transmitter, we should assign a definite string of digits to each character.

How many digits should there be in such a string?

If we assume that each word may consist of n characters and that the alphabet has N characters, then the number of likely words will be N^n, and each binary number will have

$$n \log_2 N \text{ or } [n \log_2 N] + 1$$

digits, or places. On the average, each character needs $\log_2 N$ binary digits. Therefore, $\log_2 N$ may be used as a measure of information.

Claude Shannon, one of the founders of information theory, has shown that $\log_2 N$ in probabilistic applications cannot specify information sources completely. This can be proved by comparing two information sources. One is a man tossing a coin, and the other, a man casting a die with H marked on one face and T on the remaining five. Then in both cases the alphabet may be thought of as consisting of two characters, namely H (for the head of the coin and one face of the die) and T (for the tail of the coin and the remaining faces of the die). To write down the text obtained in either case, one binary digit per character will be enough. Then $\log_2 2 = 1$, that is, the characteristic of the message source does not take into account the probabilistic behaviour of the sources.

If use is made of only the most probable texts, it can be shown on the basis of the Moivre — Laplace theorem and Stirling's formula that only H binary characters will be needed for each character of the source:

$$H = -p \log_2 p - (1 - p) \log_2 (1 - p)$$

If $p = 1/2$ (coin-tossing),

$$H = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1$$

This may be interpreted as stating that the encoding of probable messages does not cut down the number of bits per character.

If $p = 1/6$ (die-casting),

$$H = -\frac{1}{6} \log_2 \frac{1}{6} - \frac{5}{6} \log_2 \frac{5}{6} = 0.1957$$

that is, the reduction is by more than a factor of five.

Shannon has called the quantity H the *entropy of the source.*

In the more general case where a source has a finite number of distinguishable states, $A_1, A_2, \ldots, A_n$, and the probabilities of

these states are $p_1, p_2, \ldots, p_n$, then its entropy is

$$H(A) = H(p_1, p_2, \ldots, p_n) = -\sum_{k=1}^{n} p_k \log_2 p_k$$

Thus, entropy is a measure of the uncertainty of an information source and a convenient measure of information. The amount of information I about an object is equal to the difference between the a-priori (prior) and the a-posteriori (posterior) entropies of the object, or

$$I = H_{pr} - H_{post}$$

Before a signal carrying information about an object is received, only the distribution of probabilities p_k between distinguishable states is assumed to be known. The uncertainty prior to the reception of the signal may therefore be defined as the entropy

$$H = -\sum_{k} p_k \log_2 p_k$$

After reception of the signal, provided it has not been corrupted during transmission, a precise answer can be given (about the state of the object) and all uncertainty about the object is removed. In other words, the signal has reduced the uncertainty from H to 0 on the one hand and has brought with it an information I numerically equal to H; that is, *the amount of information is equal to the difference in entropy before and after the reception of a signal.*

From the example examined above it is seen that the amount of information is numerically equal to a-priori entropy, provided the information received is free from errors.

The amount of information per sequence (signal) free from errors is given by Shannon's formula

$$I = -n \sum_{i=1}^{m} p_i \log_2 p_i \tag{1.16}$$

where n is the number of symbols in the sequence, selected out of an alphabet having m symbols.

In a particular case where symbols are equiprobable and statistically independent for any n, all likely signals will likewise be equally probable, that is, $p_i = 1/m$, and the amount of informa-

tion is given by Hartley's equation:

$$I = -n \sum_{i=1}^{m} p_i \log_2 p_i = -n \sum_{i=m}^{m} \frac{1}{m} \log_2 \frac{1}{m} = n \log_2 m \quad (1.17)$$

If there are errors in the received signal, no unequivocal conclusion may be drawn about the state of the source. In other words, the entropy does not reduce to zero, and one has to take into account a-posteriori entropy in order to determine the amount of information.

Consider some properties of entropy.

1. $H(A) = 0$, if and only if any of the p_k's is unity and, as a consequence, all other probabilities are zero. This is the case when the outcome of a trial can be predicted with confidence and there is no uncertainty about it. For example, if a box holds balls all labelled with the same numeral, the probability of producing a ball is $p = 1$, and the entropy is $H(A) = -1 \log_2 1 = 0$. In all other cases, the entropy will be positive, because $p < 1$.

2. Entropy is a maximum when $p_1 = p_2 = \ldots = p_n = 1/n$, that is when the uncertainty is a maximum because there is no ground for giving preference to any one of all likely events.

Suppose there is a box holding 99 balls all labelled "1" and one ball labelled "0". Then the probability of producing a ball labelled "1" will be $p_1 = 0.99$, and that of the ball labelled "0", $p_2 = 0.01$.

The entropy for this case in binary digits will be

$$H(A) = -(0.99 \log_2 0.99 + 0.01 \log_2 0.01) = 0.081$$

If the number of balls labelled "1" is equal to that marked "0" and $p_1 = p_2 = 0.5$, then

$$H(A) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$$

The greater the uncertainty about the occurrence of events, the greater the amount of information needed about them. Conversely, the smaller the uncertainty, the smaller the amount of information needed. In the limiting case, when an event is known in advance to occur, that is, when the uncertainty about its occurrence is completely removed, $H(A) = 0$.

3. The entropy of a source AB whose states are outcomes of the joint realizations of the states A_k and B_l is equal to the sum of the entropies of the original sources A and B:

$$H(AB) = H(A) + H(B)$$

This is so because if the probability s_{kl} of the state A_kB_l for A and B statistically independent of each other is the product p_kq_l, then

$$H(AB) = -\sum_{kl} s_{kl} \log_2 s_{kl} = -\sum_{kl} p_k q_l (\log_2 p_k + \log_2 q_l)$$

$$= -\sum_k p_k \log_2 p_k \sum_l q_l - \sum_l q_l \log_2 q_l \sum_k p_k$$

Since, however, $\sum_l q_l = \sum_k p_k = 1$ by definition, then $H(AB) = H(A) + H(B)$.

Consider a case where the messages $y(t)$ treated as random processes are transmitted by means of signals $z(t)$. The amount of information about $y(t)$ contained in the signal $z(t)$ is called the *amount of mutual information* $I(y, z)$. Accordingly, the signal $z^*(t)$, received with corruption, will reproduce a message $y^*(t)$ and, as a consequence, the amount of mutual information will be $I(y^*, z^*)$.

The entropy $H(y)$ of a random process $y(t)$ is not always the same as the amount of information $I(y, z)$ contained in the process $z(t)$. In fact, if the uncertainty about the message y being transmitted cannot be removed completely on the basis of the process $z(t)$ being observed, then the uncertainty that remains may be expressed as a mutual conditional entropy, $H(y, z)$.

Hence, the amount of mutual information $I(y, z)$ may be expressed as the difference between the message entropy $H(y)$ and the mutual entropy $H(y, z)$:

$$I(y, z) = H(y) - H(y, z)$$

Owing to the symmetry of y and z, we may write

$$I(y, z) = I(z, y) = H(z) - H(z, y)$$

This is why $I(y, z) = I(z, y)$ is called the *amount of mutual information.*

Consider the entropy measure for a wider class of message sources where the probability of selecting a symbol depends on which symbols have already been selected. This probabilistic relation may, for example, occur in messages written in, say, Russian or any other particular language (where one can predict with certain probability the occurrence of a consonant following a vowel or a definite letter or word following a certain combination of letters or words).

Mathematically, stochastic processes such that the selection of a message symbol at a particular time depends on the symbols selected at the previous times are represented by Markov processes.

If the probability $p(x_k)$ of finding a symbol in the kth position solely depends on the symbol in the $(k\text{-}1)$st position, this is a *simple (singly-connected) Markov process.* If the probability $p(x_k)$ of a symbol being selected depends on the previous symbols, one has what is called a *multiple Markov process* of order n.

Thus the probability $p(x_k)$ of characters being selected out of an alphabet does not remain constant — the last n characters determine the probability $p_q(x_k)$ of the kth character being selected.

If the probability $p_q(x_k)$ is specified in advance for every next character and it is known which state is assigned to which sequence of n symbols, it is possible to compute the probability P_q for each of the states S_q $(q = 1, 2, \ldots, r)$.

For all sources of practical interest, the unconditional probability $p(x_k)$ for the kth elementary message is

$$p(x_k) = \sum_{q=1}^{r} P_q p_q(x_k) \tag{1.18}$$

The entropy of a message source in the qth state is

$$H_q = \sum_{k=1}^{n} p_q(x_k) \log_2 p_q(x_k) \tag{1.19}$$

The entropy of a message source per message character, or symbol, is

$$H(x) = -\sum_{q=1}^{r} \sum_{k=1}^{n} P_q p_q(x_k) \log_2 p_q(x_k) \tag{1.20}$$

From knowledge of the entropy, it is possible to evaluate the redundancy r_x of a message, resulting from the fact that the message elements have different probabilities:

$$r_x = (H_{\max} - H(x))/H_{\max}$$

whence

$$H_{\max} = -\sum_{k=1}^{n} \frac{1}{n} \log_2 \frac{1}{n} = \log_2 n$$

Therefore,

$$r_x = 1 - H(x)/\log_2 n$$

Going back to our example of balls labelled "1" and "0", $H(x) = 0.081$. The entropy is a maximum when $p_1 = p_2$, that is, when the balls labelled "1" are equal in number to those labelled "0". Therefore, the redundancy of the message

$$r_x = 1 - 0.081/1 = 0.919$$

Thus, the smaller the difference between p_1 and p_2, the greater the redundancy of the message.

Consider a simple Markov process. A source generates messages in an alphabet consisting of the letters A and B. The two letters have the same unconditional probability of occurrence, that is, $p(A) = p(B) = 0.5$. We also know how the probability of the letter A being selected at a given time depends on its selection at the time immediately preceding, that is, $p(A/A) = p(B/B) = 0.8$ (if the previous selection was the letter A, the probability of the letter A being selected again is 0.8). The same applies to the letter B. It is also known that $p(A/B) = p(B/A) = 0.2$ (if the previous selection was the letter A, the probability that the next selection will be the letter B is 0.2).

The unconditional probabilities $p(A)$ and $p(B)$ for the abov source are equal. This is so because on the basis of the equation for the complete probability, we may write

$$p(A) = p(A)\,p(A/A) + [1 - p(A)]\,p(A/B)$$
$$p(B) = p(B)\,p(B/B) + [1 - p(B)]\,p(B/A)$$

Since $p(A) = p(B) = 0.5$, then on the basis of (1.20) we may write

$$H = -0.5 \times 0.8 \log_2 0.8 - 0.5 \times 0.2 \log_2 0.2 - 0.5 \times 0.8 \log_2 0.8$$
$$- 0.5 \times 0.2 \log_2 0.2 = 0.722 \text{ binary unit or bit}$$

The redundancy of the message in this case is 0.278.

Consider a case where the letters A and B have different unconditional selection probabilities, that is, $p(A) \neq p(B)$ and $p(A/A) = 0.3$; $p(B/A) = 0.7$; $p(A/B) = 0.1$; and $p(B/B) = 0.9$.

The unconditional probabilities $p(A)$ and $p(B)$ can be found by the equations for the complete probability:

$$p(A) = 0.125; \qquad p(B) = 0.875$$

Then, on the basis of (1.20)

$$H = -0.125\,[0.3 \log_2 0.3 + 0.7 \log_2 0.7]$$
$$- 0.875\,[0.1 \log_2 0.1 + 0.9 \log_2 0.9] \approx 0.52 \text{ bit}$$

Hence,

$$r_x = 0.48$$

A very important characteristic of any message source is the *source rate*, $I'(x)$, defined as the amount of information generated per unit time. If a message $y(t)$ is a sequence of discrete symbols generated by the source at a rate v_x (symbols per second), then

$$I'(x) = v_x I(x)$$

where $I(x)$ is the amount of information per symbol.

1.4. ENTROPY IN THE CONTINUOUS CASE

The uncertainty about a continuous random or stochastic process can be evaluated by considering it as a limiting case.

For this purpose, we shall quantize the continuous random quantity into an even number of levels, that is, break up the entire range of likely values of x (from $-\infty$ to $+\infty$) into equal segments Δx. From comparison with the discrete case, we may note that in the continuous case the probability distribution is replaced by the probability density, $p(x^*)$, which is a dimensional quantity. To change to a dimensionless quantity (so that logarithms of dimensional quantities can be avoided), we introduce a dimensional coefficient, $1/x_0$ (where x_0 is the unit of probability density). Then,

$$x = x^*/x_0$$

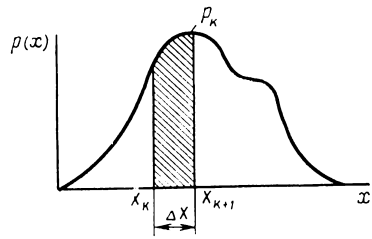

Fig. 1.8. Probability density function of a random quantity x

As a consequence, the probability density $p(x)$ will be a dimensionless function.

As a result of this quantization (Fig. 1.8), the probability density p_k (equal to the shaded area) associated with the kth segment is

$$p_k = \int_{x_k}^{x_{k+1}} p(x)\,dx \qquad (x_{k+1} = x_k + \Delta x)$$

Assuming that the continuous random quantity x is made up of discrete quantities x', we may apply an entropy measure. That is,

we may write

$$H(x') = -\sum_{-\infty}^{+\infty} p_k \log_2 p_k = -\sum_{-\infty}^{+\infty} \left[\int_{x_k}^{x_k+\Delta x_k} p(x)\,dx \right] \log_2 \left[\int_{x_k}^{x_k+\Delta x} p(x)\,dx \right]$$

Now let all $\Delta x'$s diminish. If the $\Delta x'$s are sufficiently small and $p(x)$ is sufficiently smooth, we get

$$\int_{x_k}^{x_k+\Delta x} p(x)\,dx \approx p(x_k)\,\Delta x$$

Therefore,

$$\lim_{\Delta x \to 0} H(x') \approx \lim_{\Delta x \to 0} \left\{ -\sum_{-\infty}^{\infty} p(x_k)\,\Delta x \log_2 [p(x_k)\,\Delta x] \right\}$$

$$= \lim_{\Delta x \to 0} \left\{ -\sum_{-\infty}^{\infty} p(x_k) [\log_2 p(x_k)]\,\Delta x \right\}$$

$$+ \lim_{\Delta x \to 0} \left\{ -\sum_{-\infty}^{\infty} p(x_k) [\log_2 \Delta x]\,\Delta x \right\} = \int_{-\infty}^{\infty} p(x) \log_2 p(x)\,dx$$

$$+ \lim_{\Delta x \to 0} \left\{ -[\log_2 \Delta x] \sum_{-\infty}^{\infty} p(x_k)\,\Delta x \right\}$$

$$= -\int_{-\infty}^{\infty} p(x) \log_2 p(x)\,dx - \lim_{\Delta x \to 0} \log_2 \Delta x \qquad (1.21)$$

Owing to the second term, $\lim_{\Delta x \to 0} \log_2 \Delta x$, this expression for the entropy measure of a continuous random quantity, $\Delta x \to 0$, tends to infinity. Hence we may conclude that a finite absolute measure of uncertainty cannot be applied to continuous random processes. Instead, resort is made to a relative measure of uncertainty, called the relative entropy $H_\varepsilon(x)$.

As a standard or reference, we shall use the uncertainty about the uniform distribution in an interval ε wide. On quantizing the process uniformly distributed over the interval ε, we find by analogy with (1.21) that the limit of uncertainty about the discrete quantity x'' is

$$\lim_{\Delta x \to 0} H(x'') = \log_2 \varepsilon - \lim_{\Delta x \to 0} \log_2 \Delta x$$

The relative entropy $H_\varepsilon(x)$ of a random quantity x is the limit to which the difference between the entropies of the quantized quantities x' and x'' tends, or

$$H_\varepsilon(x) = \lim_{\Delta x \to 0} [H(x') - H(x'')] \tag{1.22}$$

Expressing $H(x')$ and $H(x'')$ in accordance with (1.21) and substituting them in (1.22) gives

$$H_\varepsilon(x) = -\int_{-\infty}^{\infty} p(x) \log_2 p(x)\, dx - \log_2 \varepsilon = \int_{-\infty}^{\infty} p(x) \log_2 \varepsilon p(x)\, dx$$

This difference is finite.

Setting $\varepsilon = 1$, then

$$H_{\varepsilon=1}(x) = -\int_{-\infty}^{\infty} p(x) \log_2 p(x)\, dx$$

$H_\varepsilon(x)$ is alternately called the entropy of a continuous process and, because it is related to the differential probability distribution, also the *differential entropy*.

$H_\varepsilon(x)$ has properties which generalize in some cases those of the entropy of discrete processes.

Units of entropy. *The unit of entropy* is the uncertainty about a random process specified such that

$$H(A) = -\sum_{k=1}^{n} p_k \log_2 p_k = 1$$

For example, let us choose the least number of states for which the process still remains random ($n = 2$) and take logarithms to the base 2.

Then,

$$H(A) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 = 1$$

and, as a consequence, according to property 2 of entropy

$$p_1 = p_2 = 0.5$$

To sum up, the unit of uncertainty is the entropy measure of a random process with two equiprobable states. This unit is called the *binary unit*, or bit for short (sometimes also referred to as the binit). With a different choice of the base of the logarithms, other units can be derived. For example, taking logarithms to the base

10, we shall have a decimal unit of entropy. A process having ten equiprobable states will have an uncertainty of one decimal unit. Units for the entropy of continuous processes are defined in a similar way.

1.5. ε-ENTROPY OF DIFFERENTIABLE FUNCTIONS

In our discussion of sampling and quantizing, we have learned how a sampling or quantizing step should be specified in order to secure the desired degree of accuracy. Now we shall determine the amount of binary information necessary to reconstruct the continuous process under investigation to a specified degree of accuracy, ε. This task arises because the information generated by message sources (such as transducers) usually has a large amount of redundancy which requires communication channels and computer memories of larger capacities. This is why, before information is to be conveyed over communication channels and processed by a computer, it is important to evaluate its redundance and reduce it, if necessary.

Suppose that the functional relationship of the process under investigation is unknown, but it may be identified with a particular space (class) in advance. The quantitative measure of such a process will be the ε-entropy.

The entropy of a source of continuous random signals may be computed as follows. The space of all likely signals (whose number is infinite in the general case) is decomposed into subsets so that their number is finite. If, within each subset, we can locate a point (centre) not more than ε distant from all other points, we shall thus be able to approximate the original space accurate to within ε.

A space into which distance is introduced is called *metrical.*

If in a space of diameter d we can identify a point (centre) which is less than $d/2$ distant from all other points, the space is called *centrable.*

If a space can be decomposed into a finite number of subsets, we have a *compact* space.

Thus, a message source gives the points of a compact, centrable, metric space. Now, it is only necessary to find a minimum binary representation for the received information, from which we would be in a position to reconstruct each point of the original message (and, indeed, any process) accurate to within ε. For this purpose, we decompose the compact space A into subsets each of diameter

$d \leqslant 2\varepsilon$ (the resultant system is the ε-*range*). Then knowledge of the centre of each subset will make it possible to locate each point of the space accurate to within ε. We shall designate as N_ε the least number of subsets of diameter $d \leqslant 2\varepsilon$. Then the minimum number of binary characters in a sequence needed to record each subset (any point accurate to within ε) will be $\log_2 N_\varepsilon$, or

$$H_\varepsilon(A) = \log_2 N_\varepsilon$$

where $H_\varepsilon(A)$ is the ε-(nonprobabilistic or absolute) entropy of the metric space.

In the general case, the ε-entropy can be determined on the basis of the following theorem: for any compact set A contained in a metric space R, the following inequalities are satisfied:

$$h_{-\varepsilon}(A) \leqslant H_\varepsilon(A) \leqslant H_\varepsilon^R(A) \tag{1.23}$$

where

$$h_{2\varepsilon}(A) = \log_2 n_{2\varepsilon}(A)$$

$$H_\varepsilon^k(A) = \log_2 N_\varepsilon^n(A)$$

[$h_{2\varepsilon}(A)$ is the maximum number of points in a 2ε-distinguishable subset of space A; and $N_\varepsilon^R(A)$ is the minimum number of points in the ε-network for space A].

A 2ε-*distinguishable subset* is an ensemble of points in which the distance between any two points is greater than ε.

An ε-*network* is an ensemble of points in which it is possible to identify a point not more than ε distant from a selected point in space A.

Thus, finding the ε-entropy of a metric space reduces to constructing an ε-network and a 2ε-distinguishable set in space A. For functions having bounded derivatives of order S (such functions describe a wide range of physical processes) a general estimate of the ε-entropy is given by the Kolmogorov — Vitushkin theorem which states: for any $S \geqslant 1$, $L > 0$ and $C > 0$, there exist two positive constants A and B solely dependent on S such that, given sufficiently small $\varepsilon > 0$, the following inequalities are satisfied:

$$AT\,(L/\varepsilon)^{1/s} \leqslant H_\varepsilon(F) \leqslant BT\,(L/\varepsilon)^{1/s}$$

where F is the space of functions $f(t)$ defined on the interval $0 \leqslant t \leqslant T$ for which everywhere within this interval there exists a $(s-1)$st derivative satisfying Lipschitz' condition with a constant L, and such that $|f^{(k)}(0)| \leqslant C$ $(k = 0, 1, 2, \ldots, s-1)$.

Thus, in order to determine the ε-entropy, it is necessary to compute the constants A and B, which is done for each specific functional class separately through appropriate functional constructions (there are no unified recommendations for such constructions). How well a particular technique has been chosen will be indicated by the degree of closeness of the constants A and B.

Let us compute the ε-entropy for functions satisfying Lipschitz' condition ($s = 1$) and for twice differentiable functions ($s = 2$).

The ε-entropy for functions satisfying Lipschitz' condition. Let F designate a space which contains functions $f(t)$ belonging to the Lipschitz class and having $|f(0)| \leqslant C$, and also let Φ designate the ensemble of functions $\varphi(t)$ (Fig. 1.9) represented on the interval $(0, T)$ as t:

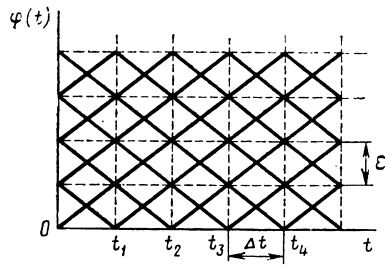

Fig. 1.9. ε-grid construction

$$\varphi(t) = \varphi(0) + L \int_0^t \psi(t)\, dt$$

where $L = \varepsilon/\Delta t$, and $\psi(t)$ is a function which can only take the values $+1$ or -1 and is constant within each segment Δt.

The numbers $|\varphi(0)| < C$ are such that two different functions, $\varphi_1(t)$ and $\varphi_2(t)$, on the interval $(0, T)$ are as follows:

$$\varphi_1(t) = \varphi_1(0) + L \int_0^t \psi(t)\, dt$$

and

$$\varphi_2(t) = \varphi_2(0) + L \int_0^t \psi(t)\, dt$$

A function defined on the interval (a, b) belongs to the first-order Lipschitz' class if $|f(t_1) - f(t_2)| \leqslant L|x_1 - x_2|$ satisfy the inequality

$$|\varphi_1(0) - \varphi_2(0)| \leqslant 2k\varepsilon \qquad (k = 0, 1, 2, \ldots, C/\varepsilon)$$

It is seen from Fig. 1.9 that the set Φ of functions $\varphi(t)$ forms an ε-network for space F. This implies that through an appropriate choice of sign for the function $\psi(t) = \pm 1$ within each segment, it

can be arranged that everywhere in the interval $(0, T)$ the following inequality will be satisfied:

$$|f(t) - \varphi(t)| \leqslant \varepsilon$$

Let us count the number of elements in the ε-network for F.

The number of values that the function can take at $t = 0$ is C/ε. The number of segments Δt within the interval $[T]$ is $T/\Delta t$. Hence, the number of functions $\varphi(t)$ forming the ε-network for space F is $(C/\varepsilon)\, 2^{T/\Delta t}$.

On the basis of (1.23), the upper bound on the ε-entropy may be expressed as

$$H_\varepsilon(F) \leqslant \log_2 (C/\varepsilon)\, 2^{T/\Delta t}$$

or

$$H_\varepsilon(F) \leqslant (T/\Delta t) + \log_2 (C/\varepsilon)$$

In order to express the lower bound on the ε-entropy, we construct 2ε-distinguishable sets in space F.

We choose some number $\eta > 0$ and put

$$\varepsilon + \eta = \varepsilon' = L' \Delta t$$

Let us show that the functions $\varphi_1'(t)$ and $\varphi_2'(t)$ expressed as

$$\varphi_1'(t) = \varphi'(0) + L' \int_0^t \psi(t)\, dt \tag{1.24}$$

are distinguishable by more than 2ε at least at one point in the interval $(0, T)$.

The numbers $|\varphi_i(0)| < C$ are such that for two distinct functions $\varphi_1'(t)$ and $\varphi_2'(t)$ they satisfy the equality

$$|\varphi_1'(0) - \varphi_2'(0)| = 2k\varepsilon' \qquad (k = 0,\ 1,\ 2,\ \ldots, C/\varepsilon')$$

If $\varphi_1'(0) \neq \varphi_2'(0)$, then

$$|\varphi_1'(0) - \varphi_2'(0)| \geqslant 2\varepsilon'$$

and

$$|\varphi_1'(0) - \varphi_2'(0)| > 2\varepsilon$$

Assume that

$$\varphi_1'(0) = \varphi_2'(0)$$

Then, noting that

$$|\psi_1(t) - \psi_2(t)| = 2\varepsilon$$

we get

$$|\varphi_1'(t_k) - \varphi_2'(t_k)| = L' \int_0^{t_k} |[\psi_1(t) - \psi_2(t)]\, dt|$$

$$= L' \int_0^{t_k} 2\, dt = 2L'\Delta t = 2L'\varepsilon'/L' = 2\varepsilon' > 2\varepsilon$$

Thus, the functions $\varphi_1'(t)$ and $\varphi_2'(t)$ are distinguishable in a 2ε-space.

It is seen from Fig. 1.9 that space F has fewer than $(C/\varepsilon')2^{T/\Delta t}$ such terms (pairwise more than 2ε distant from each other).

Therefore,

$$h_{2\varepsilon'}(F) \geqslant (T/\Delta t) + \log_2(C/\varepsilon')$$

Since the last inequality holds for any η, however small, then, on the basis of (1.23), we get

$$H_\varepsilon(F) \geqslant (TL'/\varepsilon') + \log_2(C/\varepsilon') \tag{1.25}$$

Taking account of (1.24) and (1.25) and recalling that $\Delta t = \varepsilon/L$, we have

$$(TL/\varepsilon) + \log_2(C/\varepsilon) \leqslant H_\varepsilon(F) \leqslant (TL/\varepsilon) + \log_2(C/\varepsilon)$$

i.e.

$$H_\varepsilon(F) = TL/\varepsilon + \log_2(C/\varepsilon) \tag{1.26}$$

For this case, the constants A and B in the Kolmogorov — Vitushkin theorem are equal.

The ε-entropy for twice differentiable functions. Let F designate a space which contains the functions $f(t)$ defined on the interval $(0, T)$, such that

$$\left.\begin{aligned} |f(0)| &\leqslant C \\ |f'(t)| &\leqslant N \\ |f''(t)| &\leqslant M \end{aligned}\right\}$$

where N and M are constants.

To evaluate the ε-entropy in this case, we should likewise construct an ε-network for space R and a 2ε-distinguishabe set in space F.

To construct the ε-network, we may use the family Φ of functions $\varphi(t)$ which may be given the form

$$\varphi(t) = \varphi(0) + k \int_0^t \psi(t)\, dt$$

where $k = \varepsilon/\Delta t$; $\psi(t) =$ constant within each segment Δt, $[\psi(t) = 0, \pm 1, \pm 2, \ldots, \pm(N/k + 1)]$.

The functions $\varphi(t)$ of the family Φ are such that $|\varphi(0) \leqslant C$ and $|\varphi'(t)| \leqslant N$.

Geometrically, Φ is a family of piecewise-linear continuous functions permitting each segment to have some specified number of slopes. If the function $\varphi(t)$ be fitted so as to satisfy everywhere the inequality

$$|f(t) - \varphi(t)| \leqslant (\varepsilon + \delta^2 M/8) = \varepsilon'$$

(where $\delta^2 M/8$ is the error of approximation), then Φ will be the ε-network for space F.

Once the number of elements in the ε'-network is known, it is possible, as in the previous case, to determine the upper bound on the ε-entropy.

In order to determine the lower bound on the ε-entropy, we construct a 2ε-distinguishable set in space F. Then we let G designate a set of functions $g(t)$ definable on the interval as

$$g(t) = g(0) + M \int_0^t \psi(t)\, dt$$

As a result, we obtain

$$|g(t)| \leqslant N; \qquad |g'(t)| \leqslant M$$

The function $\psi(t)$ takes only one value out of two, $+1$ or -1, within each segment Δt.

Obviously, the set of functions $f^*(t)$ given the form

$$f^*(t) = f^*(0) + \int_0^t g(t)\, dt \tag{1.27}$$

belongs to space F on the condition that $|f^*(0)| \leqslant C$. It is possible to subject the function, Eq. (1.27), to limitations such that they will form a 2ε-distinguishable set in that space.

It should be noted that an attempt to take a fuller account of apriori information about the process under investigation entails a prohibitively large amount of computational work, while the resultant refinement in the ε-entropy is practically immaterial.

To sum up, firstly, the ε-entropy defines the minimal binary sequence necessary to represent a process to a specified degree of accuracy; secondly, if a source generates information in the form of binary digits, the ε-entropy enables one to evaluate the redundancy of this information and to say whether it is advantageous to compress it.

CHAPTER 2

COMMUNICATION CHANNELS AND SYSTEMS

2.1. BASIC DEFINITIONS

Whatever its content or form, information is always conveyed from source to user. Information presented in a specific form is called a *message*. Messages are conveyed from source to user by a communication system.

A communication (or exchange) system is a collection of physical facilities and mathematical methods intended to convey or exchange messages between sources and users. In schematic form, a communication system set up to convey messages between a source and a user is shown in Fig. 2.1. It includes a transmitter, TMR, a communication medium or channel, CC, and a receiver, RVR. The *transmitter* is an arrangement of engineering facilities whose function is to convert messages generated by the source into signals which may be conveyed over the available communication medium or channel. The *communication medium* or *channel* is a collection of engineering facilities and the physical medium through which the signal is conveyed. The physical medium over which the signal (say, electromagnetic waves) is propagated is called a *line* or *link*. The *receiver* is an arrangement of engineering facilities whose function is to convert the signal appearing at the channel output back to a message.

The conversion of a message into a signal involves the operations of coding and modulation. To carry out them, the transmitter has an encoder (or coder) and a modulator. Accordingly, the receiver has a demodulator and a decoder for the reverse operations.

Communication channels may be classed in different ways. According to the service rendered, there may be telephone, television, telegraph, telemetry, telecommand, data-transmitting channels, etc. According to the physical medium used, there may be cable lines, radio-relay links, etc. According to the range of frequencies used, there may be voice-frequency, carrier-frequency, radio-frequency, microwave, and light systems.

In terms of structure, signals may be continuous, discrete, and mixed (continuous-discrete or discrete-continuous). Conti-

nuous (or analog) communication channels utilize continuous (or analog) signals; discrete (or digital) systems use discrete (or digital) signals; mixed systems make use of both. This classification of communication channels and signals leads to four combinations of information transmission from source to user.

(1) The information source generates a continuous signal conveyed to the user in continuous form; this is a continuous (or analog) communication channel.

(2) The information source generates a continuous signal which is conveyed to the user in the discrete form; this is a continuous-discrete (or analog-digital) communication channel.

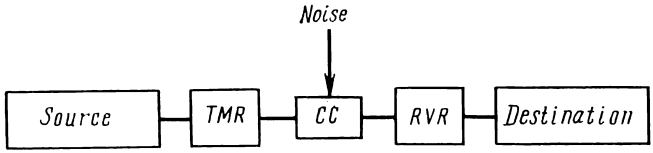

Fig. 2.1. Communication system (block diagram)

(3) The information source generates a discrete signal which is conveyed to the user in the form of a continuous function; this is a discrete-continuous (digital-analog) communication channel.

(4) The information source generates a discrete signal which reaches the user in the discrete form; this is a discrete (or digital) communication channel.

The choice of a particular channel depends on the nature of the information source, capabilities for the conversion of the primary signal generated by the source, utilization of a specific channel in a specific automatic control or management information system, and the characteristics of the user from the view-point of the signal structure.

The classification of channels into discrete (digital) and continuous (analog) is an arbitrary one, because a discrete channel will often include a continuous channel where continuous signals exist at both input and output.

In a mathematical treatment, however, the fact that a discrete channel contains a continuous channel is ignored.

Theoretically, a digital channel is defined by specifying an input alphabet of code symbols, an output alphabet of code symbols, channel capacity (amount of information transmitted by the channel per unit time), and appropriate probabilities.

According to the number, m, of code symbols used in an alphabet (that is, the number system adopted), there will be *binary* channel if $m = 2$, a ternary channel if $m = 3$, etc.

Sources and users may be interconnected either by direct (unswitched) channels or lines, or by built-up or tandem channels or lines each assembled from several channels by switching (line or channel switching) or by relaying messages received by a central system from one terminal to one or more terminals as the desired channel is made available (message switching).

The channels interconnecting terminals (sources, users) and switching centres are called *subscriber's* or *local* loops.

2.2. MODULATION AND DEMODULATION

Modulation. Through a communication system, information is conveyed by a carrier (direct or alternating current in wire lines and electromagnetic waves in radio links). At the transmitting end, the characteristics of the carrier are modified so as to establish a unique relation between the message (word or character) and the carrier variations, or signal. This transformation is called *modulation.*

Several methods of modulation are employed, each adapted to a particular form of carrier and each modifying a particular property (or properties) of the carrier. For example, a signal may use fixed states, waves, or pulses of any physical origin (acoustical, electrical, radio, Fig. 2.2).

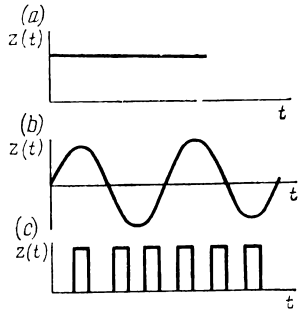

Fig. 2.2. Code signals

In Fig. 2.2*a*, the carrier $z(t)$ is in a fixed state (for example, an electric current or voltage fixed at some level). In such a case modulation modifies the state, or level, of the carrier so as to relate it in a unique way to particular code combinations (for example, 6 V for a 1 and 0.5 V for a 0).

In Fig. 2.2*b*, the carrier $z(t)$ is a sine wave. This form of carrier is utilized by frequency-division multiplex lines and radio links. The carrier characteristics or properties that can be modified for the purpose of modulation may be the amplitude V_0, the phase φ, and the frequency ω of the sine wave. The three are related such that

$$v = V_0 \cos(\omega_0 t + \varphi_0)$$

where V_0, ω_0 and φ_0 are the amplitude, frequency and initial phase (or epoch angle) of the carrier.

In Fig. 2.2*c*, the carrier $z(t)$ is a periodic train or sequence of pulses. This form of carrier is utilized in communication systems which operate by time-division multiplex. The properties or characteristics of the carrier that may be modified for the purpose of modulation are the pulse amplitude V_0, the pulse position or phase φ, pulse duration or width τ, and pulse repetition frequency f, or codes formed therefrom.

In communication systems, two methods of modulation are predominantly used, namely that of a sinewave and pulse modulation.

Modulation of a sinewave. Communication systems operating by frequency-division multiplex and radio links utilize radio-frequency (r.f.) carriers. On the other hand, the spectra of signals generated or utilized in control and digital transmission systems often lie in the audio-frequency (a.f.) range. This necessitates the use of modulation (that is, superposition of the signal wave onto the carrier wave).

Where the amplitude of the carrier sinewave is modified, amplitude modulation results. If the frequency of the carrier is varied by the signal, frequency modulation results, whereas phase modulation occurs when the phase of the carrier is altered.

Amplitude modulation. The amplitude of the carrier (r.f.) wave is modified in time in accordance with the signal (a.f.) wave (Fig. 2.3):

$$v = V_0 \cos(\omega_0 t + \varphi_0) \tag{2.1}$$

The result of such modification is an amplitude-modulated wave which can, in the general case, be expressed as

$$v_{AM} = [V_0 + \Delta v f(t)] \cos(\omega_0 t + \varphi_0)$$

or

$$v_{AM} = V_0 [1 + (\Delta v / V_0) f(t)] \cos(\omega_0 t + \varphi_0) \tag{2.2}$$

where Δv = maximum change in the amplitude of the signal (a.f.) wave

$f(t)$ = a.f. signal as a time function (modulating function)

$\Delta v / V_0 = M =$ *modulation factor*

To avoid overmodulation, the modulation factor should be less than unity ($M < 1$).

Figure 2.4 illustrates modulation where the modulating function undergoes a step change.

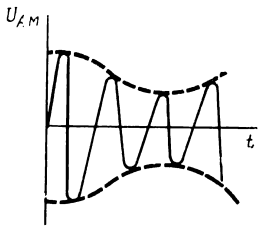

Fig. 2.3. Amplitude modulation (AM) of a sinewave signal

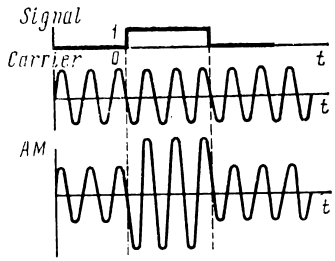

Fig. 2.4. Amplitude modulation (AM) by a step signal

If $\Delta vf(t)$ be the a.f. sinewave of frequency Ω and phase Φ, then

$$v_{AM} = V_0 [1 + M \cos(\Omega t + \Phi)] \cos(\omega_0 t + \varphi_0)$$

or

$$v_{AM} = V_0 [\cos(\omega_0 t + \varphi_0) + M \cos(\Omega t + \Phi) \cos(\omega_0 t + \varphi_0)] \quad (2.3)$$

Replacing the product of the cosines in the last term by their sum, we get

$$v_{AM} = V_0 \{\cos(\omega_0 t + \varphi_0) + (M/2) \cos[(\omega_0 + \Omega) t + \varphi_0 + \Phi] + (M/2) \cos[(\omega_0 - \Omega) t + \varphi_0 - \Phi]\} \quad (2.4)$$

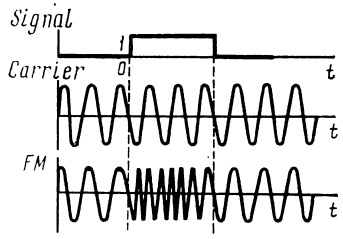

Fig. 2.5. Frequency modulation (FM)

It is seen from Eq. (2.4) that the amplitude-modulated wave is the sum of three terms: the carrier frequency ω_0 (the first term), an upper side frequency $\omega_0 + \Omega$ (the second term), and a lower side frequency $\omega_4 - \Omega$ (the third term).

Frequency modulation. Now the frequency of the r.f. carrier is varied in accordance with the a.f. signal (Fig. 2.5):

$$\omega = \omega_0 + \Delta\omega f(t) \quad (2.5)$$

where $\Delta\omega$ is the frequency deviation, or the peak difference between the instantaneous frequency of the modulated wave and its carrier frequency, and $f(t)$ is a time function of the a.f. signal.

The above expression cannot be substituted directly in

$$v = V_0 \cos(\omega_0 t + \varphi_0)$$

as was done in the previous case, because a change in frequency is always accompanied by a change in phase and vice versa. That is, frequency and phase modulations are interrelated (sometimes they are referred to by the common name of *angle modulation*).

Suppose that

$$\Delta\omega f(t) = \Delta\omega_m \cos(\Omega t + \Phi) \tag{2.6}$$

Substituting (2.6) in (2.5) yields

$$\omega = \omega_0 + \Delta\omega_m \cos(\Omega t + \Phi)$$

If the carrier be an alternating voltage highly stable in amplitude such that

$$v = V_0 \cos\theta(t) \tag{2.7}$$

(where $\theta(t) = \omega_0 t + \varphi_0$ is the instantaneous phase of the a.c. voltage), then the frequency will be varying and its instantaneous value will be

$$\omega = d\theta/dt$$

and the phase of a wave having a variable angular velocity

$$\theta = \int_0^t \omega\, dt \tag{2.8}$$

Because of this, we may re-write (2.7) as

$$v = V_0 \cos \int_0^t \omega\, dt$$

Thus, to derive an expression for frequency modulation, we should integrate (2.5)

$$\int_0^t \omega\, dt = \int_0^t [\omega_0 + \Delta\omega f(t)]\, dt \tag{2.9}$$

Thence, noting (2.8),

$$\theta = \omega_0 t + \Delta\omega \int_0^t f(t)\, dt \tag{2.10}$$

Designating $\int_0^t f(t)\,dt = F(t)$, we may write

$$v_{FM} = V_0 \cos[\omega_0 t + \Delta\omega F(t)] \tag{2.11}$$

If $f(t)$ is a sinewave of frequency Ω and phase Φ, then

$$v_{FM} = V_0 \cos\left[\omega_0 t + \Delta\omega \int_0^t \cos(\Omega t + \Phi)\,dt\right]$$

$$= V_0 \cos[\omega_0 t + (\Delta\omega/\Omega)\sin(\Omega t + \Phi) + \varphi_0]$$

The ratio $\Delta\omega/\Omega = m$ is called the modulation index and has the sense of the peak difference between the instantaneous phase of the modulated wave and its carrier phase:

$$v_{FM} = V_0 \cos[\omega_0 t + m\sin(\Omega t + \Phi) + \varphi_0]$$

If the modulating function $f(t)$ is a more complex wave which can be represented by a sum of cosinusoids, then

$$v_{FM} = V_0 \cos\left[\omega_0 t + \sum_{k=1}^{n} m_k \sin(\Omega_k t + \Phi_k) + \varphi_0\right] \tag{2.12}$$

where $m_k = \Delta m_k/\Omega_k$ are the modulation indexes dependent on the amplitudes and frequencies of the respective harmonics.

Phase modulation. In this method ot modulation, the angle of the sinewave carrier deviates from its original (no-signal) angle by an amount proportional to the instantaneous value of the modulating wave, that is, the signal being transmitted (Fig. 2.6):

$$\varphi = \varphi_0 + \Delta\varphi f(t) \tag{2.13}$$

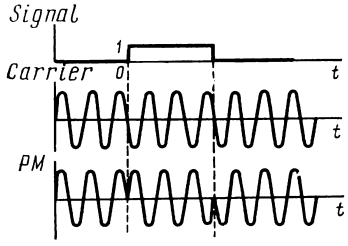

Fig. 2.6. Phase modulation (PM)

where $\Delta\varphi$ is the phase deviation defined as the peak difference between the instantaneous angle of the modulated wave and the angle of the carrier, due to modulation by the a.f. signal, and $f(t)$ is the a.f. signal function.

An analytical expression for a phase-modulated wave can be derived by substituting (2.13) in the equation for a sine wave:

$$v_{PM} = V_0 \cos[\omega_0 t + \Delta\varphi f(t)] \tag{2.14}$$

where $\varphi_0 = 0$. From comparison of (2.14) and (2.11) it is seen that the argument of the cosine contains the integral of the modulating function, $\int_0^t f(t)\,dt$, in the case of frequency modulation, and the modulating function itself, $f(t)$, in the case of phase modulation. If the modulating (signal) function is a sinusoid of the form $f(t) = \sin \Omega t$ where the frequency Ω is fixed, it is difficult to differentiate between frequency- and phase-modulated waves. In contrast, it is an easy matter to tell a phase-modulated wave from a frequency-modulated one in the case of a more complex modulating function with a time-varying frequency, $\Omega(t)$.

Pulse modulation. In pulsed communication systems, the carrier is a sequence of pulses with a duration τ_m considerably shorter than their repetition period T_0. The greater the pulse duty cycle, that is, the ratio of pulse period to pulse duration, $\gamma = T_0/\tau_m$, the smaller the energy of the pulse signal in comparison with that of a continuous signal (assuming the same peak value in either case). The increased spacing between pulses may be utilized for the insertion of pulses belonging to other channels, that is, for time-division multiplex working.

The pulse carrier may be defined by the following expression:

$$v = V_0 \sum_i f_1(t - t_0 - iT_0) \qquad (i = 0,\ 1,\ 2,\ \ldots) \qquad (2.15)$$

where V_0 = pulse amplitude
$f_1(t)$ = function describing the waveform of a single pulse
t_0 = initial phase (shift relative to a selected reference)
T_0 = pulse repetition period

Most often, a sequence of unidirectional (that is, single-polarity) rectangular or square pulses may be completely defined by specifying four characteristics: V_0, t_0, T_0 and τ_p (where τ_p is the pulse duration).

In *pulse modulation*, some characteristic(s) of the pulse chain carrier is modified in accordance with the signal to be transmitted. According to the pulse characteristic thus modified, there may be pulse-amplitude modulation (PAM), pulse-width or pulse-duration modulation (PWM or PDM), pulse-position modulation (PPM), and pulse-frequency modulation (PFM).

Consider each type of pulse modulation in greater detail.

Pulse-amplitude modulation (PAM). In pulse-amplitude modulation, the signal (information) is embedded in the amplitude of

the carrier pulses only. The PAM signal may be defined as

$$V(t) = [V_0 + \Delta v f(t)] \sum_i f_1(t - t_0 - iT_0)$$

$$= V_0 [1 + (\Delta v / V_0) f(t)] \sum_i f_1(t - t_0 - iT_0) \qquad (2.16)$$

PAM may be of one of two types. In one (Fig. 2.7*a*) the amplitude of each pulse is varied in accordance with the modulating function; in the other (Fig. 2.7*b*) the amplitude of each pulse remains unchanged and equal to the modulating function at some instant during the existence of the pulse (for example, to the

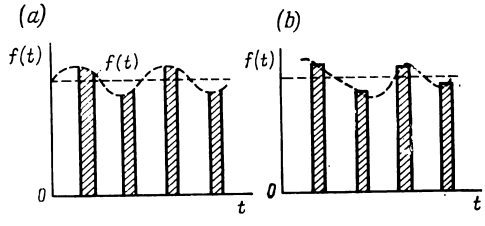

Fig. 2.7. Pulse-amplitude modulation (PAM)

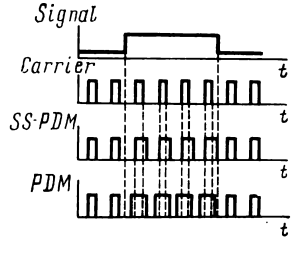

Fig. 2.8. Pulse-duration modulation (PDM)

modulating function at the start of the pulse). The general nature of variations in the pulse-chain carrier follows the modulating function.

Pulse-duration modulation (PDM). In this method, the information (signal) modulates the duration of the pulse. Since its leading and trailing edges are caused to shift by the modulating signal (Fig. 2.8) across the pulse width this method of modulation is also called *pulse-width modulation*, PWM. A modification of the technique is one in which only the position of, say, the leading or trailing edge is changed. Suppose that the unmodulated carrier is a chain of rectangular video pulses of duration τ_p. The point of reference will be the middle of the pulse. Then, in the interval equal to the repetition period T_0, the modulated carrier will be described as

$$f_1(t) = \begin{cases} V_0 & t_1 < t < t_2 \\ 0 & t_1 > t > t_2 \end{cases} \qquad (2.17)$$

where t_1 and t_2 are the times defining the positions of the leading and trailing edges of the pulse, and $\tau_p = t_2 - t_1$.

Suppose that pulse-duration modulation by a modulating function of the form $f(t) = \sin \Omega t$ changes only the position of the leading edge, while that of the trailing edge remains unaffected (one-sided PDM).

Let the peak deviation of the leading edge be $\Delta\tau$ (such that $\Delta\tau \leqslant \tau_p$). Then the position of the pulse edges in the PDM carrier will be defined as

$$t_1(t) = -0.5\tau_p - \Delta\tau \sin \Omega t$$
$$t_2(t) = 0.5\tau_p$$

Expand the unmodulated carrier chain of video pulses into a Fourier series

$$v(t) = \frac{V_0(t_2 - t_1)}{T_0} + \frac{V_0}{\pi} \sum_{k=1}^{\infty} \frac{1}{k} [\sin k\Omega_0(t - t_1) - \sin k\Omega_0(t - t_2)] \quad (2.18)$$

Substituting appropriate expressions for t_1 and t_2 gives an expression for the modulated carrier

$$v_{\text{PDM}}(t) = (V_0/T_0)(0.5\tau_p + 0.5\tau_p + \Delta\tau \sin \Omega t)$$
$$+ (V_0/\pi) \sum_{k=1}^{\infty} (1/k) [\sin k\Omega_0(t + 0.5\tau_p + \Delta\tau \sin \Omega t) - \sin k\Omega_0(t - 0.5\tau_p)]$$

whence

$$v_{\text{PDM}}(t) = (V_0/T_0)\tau_p + (V_0/T_0)\Delta\tau \sin \Omega t$$
$$+ (V_0/\pi) \sum_{k=1}^{\infty} (1/k) [\sin k\Omega_0(t + 0.5\tau_p + \Delta\tau \sin \Omega t) - \sin k\Omega_0(t - 0.5\tau_p)] \quad (2.19)$$

Or, in words, in the course of modulation the coefficients of the Fourier series vary periodically with time rather than remain constant. The spectrum of a PDM signal is more elaborate than that of a PAM signal for the same form of the modulating function because of the distortion caused by modulation.

Pulse-position modulation. The information (signal) modulates the time position of the pulse in relation to a reference time (Fig. 2.9). Pulse duration remains unaffected.

Let the modulating (information) function have the form

$$f(t) = \sin \Omega t$$

Suppose that at $\sin \Omega t > 0$ the carrier pulses are shifted to the left relative to the reference time (that is, the pulses are advanced) and that at $\sin \Omega t < 0$ they are shifted to the right (that is, the pulses are delayed). Then the change in the time position of the pulse edges for pulse-position modulation may be defined as

$$t_1(t) = 0.5\tau_p - \Delta t \sin \Omega t \qquad (2.20)$$
$$t_2(t) = 0.5\tau_p - \Delta t \sin \Omega (t - \Delta t)$$

where Δt is the time shift of the pulse.

Substituting the above expressions in that for an unmodulated chain of rectangular video pulses expanded into a Fourier series yields

$$v(t) = (V_0/T)[0.5\tau_p - \Delta t \sin \Omega (t - \Delta t) - 0.5\tau_p + \Delta t \sin \Omega t]$$
$$+ (V_0/\pi) \sum_{k=1}^{\infty} (1/k)[\sin k\Omega_0 (t - 0.5\tau_p + \Delta t \sin \Omega t)]$$
$$- \sin k\Omega_0 [t - 0.5\tau_p - \Delta t \sin \Omega (t - \Delta t)] \qquad (2.21)$$

It is seen from (2.21) that modulation causes the coefficients of the Fourier series to change. Analysis of the signal-spectrum structure reveals frequency distortion. This is why a PPM signal is demodulated by first converting it into a PDM signal from which the message spectrum is separated by a low-pass filter.

Pulse-frequency modulation (PFM). Here, the information modulates the frequency of the transmitted pulse.

The difference between PP and PF modulation is about the same as between the phase and frequency modulation of a sine wave carrier.

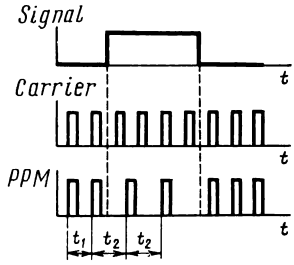

Fig. 2.9. Pulse-position modulation (PPM)

Demodulation. This is the process of recovering the original (modulating) signal from the modulated carrier. With continuous modulation, demodulation is in principle identical with signal detection which involves two steps, namely the generation of the original a.f. signal on the basis of the modulated r.f. carrier and the separation of the desired a.f. signal from the r.f. carrier.

The first step is accomplished by some type of detector, and the second, by filters.

AM signals are often detected by push-pull detectors, because of which the full-wave rectification of the carrier is obtained. For example, if the modulating function $f(t)$ be $\sin \Omega t$ and $\varphi_0 = 0$, then, on the basis of (2.2) for the AM voltage wave we have

$$v_{AM} = V_0 (1 + M \sin \Omega t) \sin \omega_0 t \tag{2.22}$$

The current in a linear detector, $i_{AM} = k|v_{AM}|$, can be found by taking the magnitude of (2.22). Since the magnitude of a product is equal to the product of the magnitudes of the multipler and the multiplicand, one has to find the magnitude of each factor. Therefore, noting that the first two terms are always positive, the current may be expressed as

$$i_{AM} = kV_0 (1 + M \sin \Omega t) |\sin \omega_0 t| \tag{2.23}$$

where $|\sin \omega_0 t|$ is the rectified sine wave in the case of full-wave rectification.

Expanding $|\sin \omega_0 t|$ into a Fourier series yields

$$|\sin \omega_0 t| = (2/\pi) \left(1 - 2 \sum_{k=1}^{\infty} \frac{\cos 2k\omega_0 t}{4k^2 - 1} \right)$$

Therefore,

$$i_{AM} = (2/\pi) kV_0 \left[(1 + M \sin \Omega t) - 2 \sum_{k=1}^{\infty} \frac{\cos 2k\omega_0 t}{4k^2 - 1} - 2M \sum_{k=1}^{\infty} \frac{\sin \Omega t \cos 2k\omega_0 t}{4k^2 - 1} \right] \tag{2.24}$$

It is seen from (2.24) that the detected current contains the original or modulating signal, $1 + M \sin \Omega t$, which is the objective of demodulation, and carrier harmonics at frequencies $2k\omega_0$ (the first summation) and at frequencies $2k\omega_0 + \Omega$ (the second summation). If a detector has a nonlinear response, use may only be made of the nonlinear relations symmetrical about the y-axis. Relations of this form should contain an even part. For example, circuits having a response which contains the functions $|x|$, x^2, x^4, $\cos x$, etc. can be used as detectors, and those with a response containing the functions $|x|$, x^3, $\sin x$, etc., cannot.

In the demodulation of an FM signal, the latter is first converted to an AM wave which is then detected as already explained. The detection of a PAM signal does not practically differ from that of an AM signal, except that the r.f. component must likewise be filtered out. The same applies to PDM signals because after a

pulse-modulated signal has been detected in the usual manner, the a.f. component is proportional to the area of the pulse, and it is immaterial whether modulation changes the amplitude or duration of the pulse envelope.

In the case of PPM, the pulse area remains unchanged and modulation only causes the pulse to shift relative to a reference (clock) time. This is why, in demodulating PPM signals the latter should first be converted to a PAM or a PDM signal or to a form where the pulse area is varied. This conversion can be accomplished by a flip-flop set by clock pulses and reset by PPM pulses.

2.3. INTERFERENCE AND NOISE

Because of imperfections in communication channels, transmitters, and receivers, and also because of external disturbances, the signal $z(t)$ is distorted and the information transmitted is mutilated.

In a continuous (analog) channel, interference and noise bring about changes in the waveform, scale and delay of the received signal relative to the transmitted one. In discrete (digital) channels, the same factors result in the distortion of the signals transmitted or in the appearance of false signals.

If the distorted signal $z'(t)$ at the output is connected to the input signal $z(t)$ by a definite functional relation which enables the original signal to be completely reconstructed, we have *regular* distortion.

If, on the other hand, noise and interference destroy one-to-one correspondence between the signals at input and output, we have *irregular* distortion.

According as the disturbance $\xi(t)$ acts on the signal there may be *additive* noise $\xi_a(t)$ or *multiplicative* noise $\xi_m(t)$.

In the case of *additive noise*, the received signal $z'(t)$ will be the sum of the transmitted signal $z(t)$ and disturbance

$$z'(t) = z(t) + \xi_a(t) \tag{2.25}$$

In the case of *multiplicative noise*, the received signal may only be expressed as the product of the transmitted signal and disturbance:

$$z'(t) = z(t)\,\xi_m(t) \tag{2.26}$$

Noise may come from a multitude of sources, and it is practically impossible to take account of the vagaries of each. Because of this, all kinds of noise are treated statistically, by reducing them

to a few types. The latter may be classed into fluctuation noise and impulse noise.

Fluctuation noise is a continuous random function of time. This function can describe a wide class of noise originating in a great multitude of (mainly man-made) sources. A distinction of fluctuation noise is that it displays the superposition of a great number of transients, the presence of spikes and the practical absence of spikes exceeding the mean level more than three or four times. Since it appears to be uniformly distributed across the frequency spectrum, it is also referred to as *continuous noise.* It is also called *white noise*, taken from the analogous definition of white light.

In the statistical theory of stochastic processes it is proved that when a great number of individual processes are superimposed on one another, their sum distribution is very close to normal, or Gaussian. Hence, fluctuation noise for the most part obeys normal or Gaussian distribution.

Impulse noise consists of sequences of pulses varying in duration and intensity and with a more or less random occurrence in time. This form of noise does not display the superposition of individual transients or spikes. Impulse noise originates in man-made sources (car ignition systems, power lines, welding equipment).

Often, impulse noise is *random impulse noise.* The associated disturbances are most severe in the case of impulse noise having characteristics or properties very close to those of useful signal and arising in receivers operating on the threshold principle.

2.4. CAPACITY OF A COMMUNICATION CHANNEL

Consider a communication channel whose input accepts a multitude of signals $z(t)$ and whose output delivers a multitude of signals $z'(t)$.

The signal $z(t)$ is used to convey a message $y(t)$ over the channel. The amount of information $I'(y, z)$ about $Y(t)$ contained in the transmitted signal $z(t)$ and presented to the channel input per second is defined as the *rate of input to the communication channel.*

The amount of information $I(y, z')$ about $y(t)$ contained in the received signal $z'(t)$ and transmitted per second is defined as the *rate of transmission over the communication channel.*

Suppose that the signal $z(t)$ at the channel input is a true representation of input information, that is, $Y(y, z) = Y(y)$ and similarly at the channel output, $Y(y, z') = Y(z, z')$.

If the rate at which the signal $z'(t)$ is transmitted over the channel is V_z, then the rate of information transmission is

$$Y'(z, z') = V_z, Y(z, z')$$

where $Y(z, z')$ is the amount of information per symbol at the channel output.

Since $I(y, z) = H(y) - H(y, z)$, we may write

$$Y(z, z') = H(z) - H(z/z') = H(z') - H(z'/z) \qquad (2.27)$$

where $H(z)$ = entropy per symbol for the input signal
$H(z')$ = entropy per symbol for the output signal
$H(z/z')$ = conditional (mutual) entropy per symbol in the input and output signals

Noise acts to distort the discrete sequence of symbols conveyed over a real communication channel so that the symbol y'_1 in some ith position at the receiving end will differ from the transmitted symbol y_i.

In an elementary case, the conditional probabilities $p(y'/y)$ that the channel output will deliver a symbol y' in the ith position for the symbol y applied to the channel input will be the same for all ith positions and independent of y' and y in any other positions. In such a case, the discrete communication channel is called a *discrete constant channel without memory*, or a *uniform channel.*

If, on the other hand, these probabilities depend on time or the previous events, we have a *channel with memory*, or a *nonuniform channel.*

In a simple case, a discrete channel may be taken to be a *symmetric channel without memory.* By definition, this is a discrete constant channel in which all symbols have the same probability of being mutilated, and the fidelity of transmission does not depend on the statistics of the sequence being transmitted. Then the effect of a disturbance may be visualized as the digit-by-digit addition of the input sequence of code symbols y_i and a sequence of interfering symbols E_i, the latter being independent of the former and generated by an assumed noise source whose statistics completely define the channel.

In a symmetric channel without memory errors of multiplicity t obey the binomial distribution. Designating the error probability of a symbol as p and the probability of correct reception for the same symbol as $q = 1 - p$, the error probability distribution of a

code combination made up of n symbols can be defined by expanding the binomial into a series of the form:

$$(p+q)^n = q^n + C_n^1 p^1 q^{n-1} + C_n^2 p^2 q^{n-2} + \ \ldots \ + p^n = \sum_{t=0}^{n} C_n^t p^t q^{n-t} \tag{2.28}$$

Each term in the series, Eq. (2.28), characterizes the probability of occurrence of a t-uple error in the combination for the specified error probability p of the symbols.

Of greater practical interest are symmetric channels with memory where the transition probabilities $p(y'/y)$ for each pair ij depend on both time and the transitions that have occurred earlier.

In a symmetric channel with memory, the error distribution within a block of symbols of any length n is not always binomial. An overwhelming number of real channels tend towards a multilevel grouping of errors; that is, a channel tends to store some particular state.

When error grouping is described in terms of a simple Markov process, a channel is represented by a set S_i of states which can change into one another with a probability p_{ij} and where errors are independent and occur with a probability p_i. The simplest model of this type is the Gilbert model. In this model, the channel may reside in states S_1 and S_2 such that no errors occur in state S_1 and $p_1 = 0$, while in state S_2 the error probability is p_2. If the probabilities of direct and reverse transitions, p_{12} and p_{21}, and also the probability of preserving the original states, p_{11} and p_{22}, are known, then the error statistic is a simple Markov chain of states and is defined by a matrix

$$\mathbf{P}\,(S) = \left\| \begin{matrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{matrix} \right\| \tag{2.29}$$

In order to satisfy the condition for errors to be grouped into bursts, the transition probabilities should be considerably smaller than the state-preserving probabilities, that is,

$$p_{12} \ll p_{11}; \qquad p_{21} \ll p_{22}$$

The probabilities that the channel will reside in state S_1 or S_2 respectively are

$$P_1 = p_{21}/(p_{12} + p_{21})$$

and

$$P_2 = p_{12}/(p_{12} + p_{21})$$

and the probability of error in a symbol is

$$p = P_2 \times p_2 = P_2 \times p_{12}/(p_{12} + p_{21})$$

Since p is ordinarily a small fraction of p_2, so $p_{12} < p_{21}$.

A simple generalization of Gilbert's model is one allowing errors in both the good and the bad state. For such a model, the error statistics will be defined by a matrix of transition probabilities $\mathbf{P}(S)$ of the form of (2.29) and by two probabilities, p_1 and p_2, while the probability of error in a symbol is

$$p = p_1 \times P_1 + p_2 \times P_2 = \frac{p_1 \times p_{21} + p_2 \times p_{12}}{p_{21} + p_{12}}$$

The other properties remain the same as they are in Gilbert's model, because the matrix of transition probabilities, $\mathbf{P}(S)$, for the two models is the same.

A disadvantage of the above models is that they fail to define the instant of error-burst formation and a real distribution of error-burst length. Yet, they are a valuable tool in analysis of discrete data transmission systems.

The maximum rate at which information can be transmitted for all permissible distributions of input signals is called the *channel capacity* and symbolized as C:

$$C = \max I(z, z')$$

If the channel capacity per symbol

$$C_1 = \max H(z, z')$$

is known, then

$$C = V_z C_1$$

In all practical cases, channel capacity is finite.

In his encoding theorem, Shannon relates channel capacity and source as follows: *For any discrete message source, the source's output can be encoded in channel signals $z(t)$ and reconstructed from channel output signals $z'(t)$ with a probability of error however close to zero if the message rate is less than channel capacity.*

For a distortionfree channel, $y(t) = y'(t)$, the amount of information in the received symbols is the same as the source entropy:

$$I(y, y') = H(y) - H(y/y') = H(y)$$

By definition, the maximum channel capacity is max $I'(y, y')$. Since, however, the channel is noiseless, $I'(y, y')$ can be reached only when the message source of entropy $H_{\max}(y)$ generates symbols at a rate $V_{\max}$. However, if any symbols out of an alphabet of size m have the same probability of being selected, $H_{\max}(y)$ will be

$$H_{\max} = \log_2 m$$

with

$$p_1 = p_2 = \ldots = p_m = 1/m$$

Thus the capacity of a noiseless channel is

$$C = V \log_2 m$$

The capacity of a symmetric channel decreases with increasing entropy $H(E)$ of the noise source:

$$C_1 = \log_2 m - H(E)$$

For a binary symmetric channel $m = 2$, and so channel capacity in the absence of noise is $C = V$. That is, channel capacity is numerically equal to the rate at which information comes from the message source.

2.5. STRUCTURE AND ORGANIZATION OF INFORMATION NETWORKS

An arrangement of engineering facilities within a communication system which are intended to transmit information is called a *communication (exchange) network.*

Among the networks specifically designed to handle only one type of information are telegraph networks, telephone networks, digital data transmission networks, etc. In turn, these networks are classed into local, integrated, nation-wide and global.

A *local communication network* (Fig. 2.10) has an information-computer office, *ICO*, and a number of users' stations, *US*, linked to the *ICO* by communication channels. Where use is made of unswitched channels, they are interfaced to the *ICO* by a multiplexer, *M*. Where the channels are switched, connections are set up over a telephone or a telegraph network via appropriate automatic telephone exchanges, *ATX*, or automatic teleprinter exchange service, *TEX*. All information, processed, stored or generated by a local network remains enclosed there.

An *integrated communication network* (Fig. 2.11), in contrast to a local network, has several information-computer offices and their respective user stations, *US*. Therefore, it also incorporates local and interoffice networks.

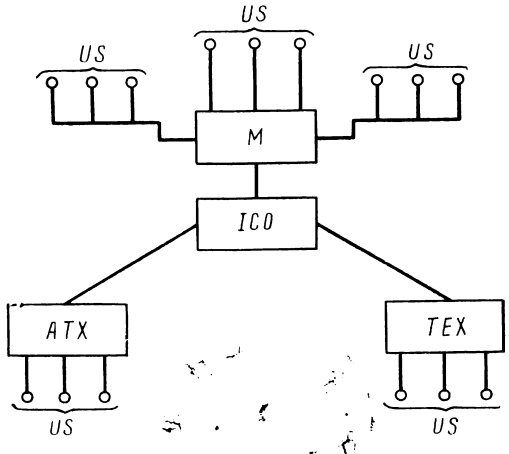

Fig. 2.10. Local communication network

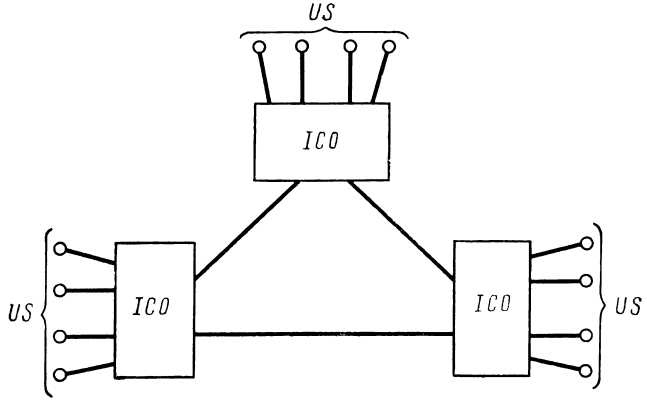

Fig. 2.11. Integrated communication network

A *nation-wide network*, *NWN*, and a *global network*, *GN* (Fig. 2.12) are set up on the basis of dispersed management information systems, *MIS*, and control systems. The users of a nation-wide or global network are the individual management-information systems (firms, industries, ministries, etc.). They are

combined into a single nation-wide or global information utility via message-switching centres, *MSC*, and the chief information-computer office, *CICO*.

Nation-wide and global networks are highly reliable and have maximum capacity, because each *MSC* is connected to its neighbours by at least two unswitched channels.

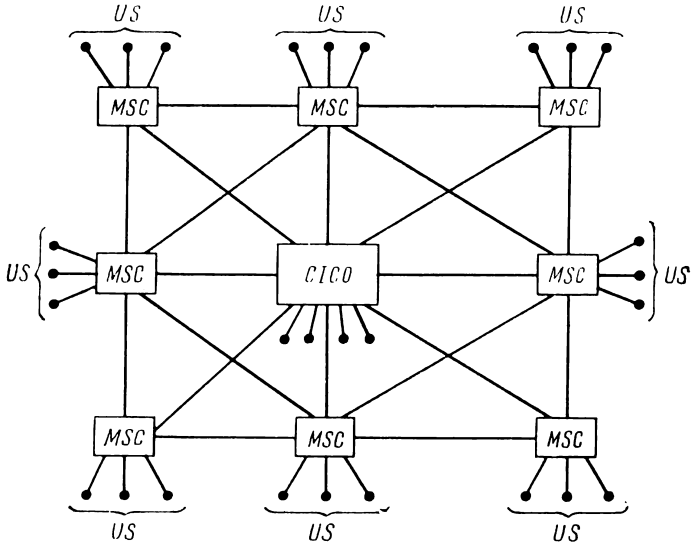

Fig. 2.12. Nation-wide exchange network

User networks are radial in structure and link their users to a computer via a multiplexer or simply a human operator. Most often, the communicating medium is in the form of intracity or district telephone and telegraph lines.

Interoffice networks are set up by placing unswitched channels between switching centres and *ICO*s. Connections may be of the "each-to-each" type (Fig. 2.13*a*), radial-nodal type (Fig. 2.13*b*), radial-ring type (Fig. 2.13*c*), and grid type (Fig. 2.13*d*).

The basic type of link used in interoffice networks is the unswitched duplex telephone channels of multichannel communication systems. Exchange of information within interoffice networks is fully automated.

As already noted in passing, information within communication networks can be exchanged in one of two ways, namely:

(1) by channel, or line (circuit) switching, where channels are switched so as to complete a continuous path for exchange of information from one terminal via switching centres to another terminal;

(2) by message switching, where messages are first received by a central system from one terminal and then sent to their destina-

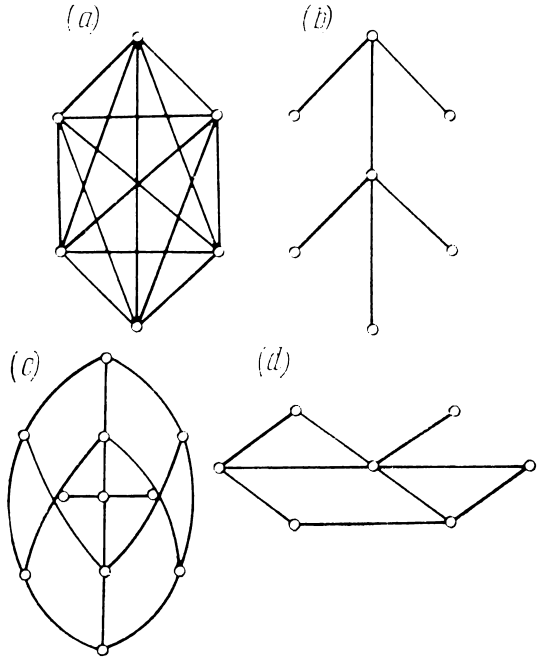

Fig. 2.13. Exchange-network geometries

tions in a kind of relay as channels in the desired direction become available.

With channel switching, the user wishing to send information requests an appropriate switching centre to set up connection in a particular direction. Following this request, the switching centre completes a "through" path by selecting one or several channels linking together the calling and the called parties. The connection is cleared only after all information has been exchanged. Should the channel(s) in the direction of the wanted party be engaged, the call is lost, and the user will have to re-apply for connection. This is why line-switching networks are also called *lost-call systems.*

With message switching, information is transmitted by relay from the source to the nearest switching centre where it is stored along with its destination address. Should a channel somewhere on the way to destination be engaged, the message is put on the queue and the service is delayed until the channel is cleared. For this reason, message-switching networks are also called *delayed-service systems.*

Because line-switching networks have no storage facilities, an increase in traffic above the average loading leads to a sudden increase in the lost-call probability and, as a consequence, to an avalanche growth of repeated calls.

Storage facilities in message-switching systems make the channels completely independent of traffic which can be re-distributed in an optimal manner. This is why a message-switching system can handle twice or three times as many messages as a line-switching system.

A line-switching network can only link together sources and users employing the same communicating media and operating at the same information-generating rates. A message-switching network can exchange information between users widely differing in information-generating rates and message format, because the various sections of the exchange path are completely independent of one another as regards the processing and relaying of information. Also, since it is a fully automated system, a message-switching network may give pre-emptive service to messages according to their priority rank.

As a rule, a message-switching network has an integrated structure, and a line-switching network, a radial structure. The network structure should have a certain redundance in terms of channels so that bypass routes can be established, should some channels or switching centres fail or be overloaded. Structural redundance enables the network to adapt itself to continually changing conditions of information exchange. Channel redundance may be provided by duplicating individual channels between neighbouring switching centres or through overall duplication arising from the expansion of the entire exchange network.

A very important indicator of how the structure of a network is efficient is the probability $Q(G_{ij})$ for the existence of a path between any pair of switching centres, G_i and G_j. This probability is defined by the product of the availability coefficients of the channels, Q_g, and of the switching centres, Q_G. As a rule, Q_g is a function of channel length and ranges between 0.7 and 0.97, while

Q_G depends on the reliability of switching-centre plant and is 0.99-0.999.

In the case of channel duplication, the minimum probability $Q(G_{ij})$ is given by

$$Q_c(G_{ij}) = \left\{ \prod_{G=1}^{G-1} Q_i \prod_{G=1}^{G} \left[1 - \prod_{g=1}^{g+1} (1 - Q_g) \right] \right\} Q(G_i) Q(G_j) \quad (2.30)$$

where G is the number of through-traffic centres and g is the number of duplicate (stand-by) channels.

With overall duplication,

$$Q_0(G_{ij}) = \left[1 - \prod_{c=1}^{c=\beta} \left(1 - \prod_{c=1}^{c-1} Q_i \prod_{g=1}^{g} Q_g \right) \right] Q(G_i) Q(G_j) \quad (2.31)$$

where β is the number of alternate paths.

It follows from (2.30) and (2.31) that for an effective increase in the probability of path existence, $Q(G_{ij})$, structural redundance should first be augmented through channel duplication and only then by overall duplication. For example, model tests have shown that radial structures always have a lower $Q(G_{ij})$ than radial-ring structures using the same number of channels. Also, in the choice of a network structure it should be sought to make the switching centres operate more reliably than the channels, because even a minute drop in the availability coefficient Q_G will drastically cut down the value of $Q(G_{ij})$ even for highly reliable channels.

The rapid growth of information-exchange systems adds urgency to improvements in the networks on which they are based. At present, there is a tendency towards a wider use of channel- and message-switching networks built specifically for information exchange apart from conventional communications facilities. The new networks, especially of the message-switching type as more efficient, will improve the utilization of large data-processing and computer centres and prevent impairment in the services rendered by telephone and telegraph channels.

CHAPTER 3

CODES AND ERROR CORRECTION

3.0. DEFINITIONS

A *code* is a system of characters and rules used to represent information for the purpose of transmission or storage. A code character is a combination of elements x_i taken out of an agreed or prescribed alphabet. The total number of elements or symbols in an alphabet may be written in any number system to the base m. According to the value of the base, codes may be binary if $m = 2$, ternary if $m = 3$, etc. An n-unit code to base m in which all of m^n code combinations (or at least more than m^{n-1} combinations) are used is a *primary* or *simple* code.

If each combination is of the same length as measured in unit elements or in time duration, this is a *uniform-* or *equal-length* code; if not, this is a *nonuniform-* or *unequal-length* code.

Nonuniform codes reduce the total number of symbols needed for the transmission of messages. For example, the transmission of the decimal digits 0, 1, 2, ..., 9 by means of binary tetrads would require a total of 40 binary symbols for the ten decimal digits. In a nonuniform code such as

$$0 \rightarrow 0, \quad 1 \rightarrow 1, \quad 2 \rightarrow 10, \quad 3 \rightarrow 11, \quad 4 \rightarrow 100,$$
$$5 \rightarrow 101, \quad 6 \rightarrow 110, \quad 7 \rightarrow 111, \quad 8 \rightarrow 1000, \quad 9 \rightarrow 1001$$

the total number of binary symbols would be cut down to 28. A drawback of nonuniform codes, which has limited their use, is that they are difficult to decode automatically and, especially, to print out in alphameric form.

Primary codes appeared along with the early telegraph systems.

3.1. TELEGRAPH CODES

Of the primary codes, the most typical one is the Morse code first proposed by Samuel F.B. Morse of the United States in the mid-19th century. In its early form, decimal digits were encoded as so many serrations on a waveform, with an inverted serration for zero.

As it is used today, the Morse code is a system of dots and dashes (Fig. 3.1) arranged so as to assign the shortest combinations to the most frequently used letters. For example, dot-dash is the Morse code character for the letter *A*, a dot for *E*, dot-dot for *I*, dash-dash for *M*, dash-dot for *N*, a dash for *T*, etc. Since all decimal numerals have the same probability of occurrence, they are all represented by five-unit code characters. The longest code

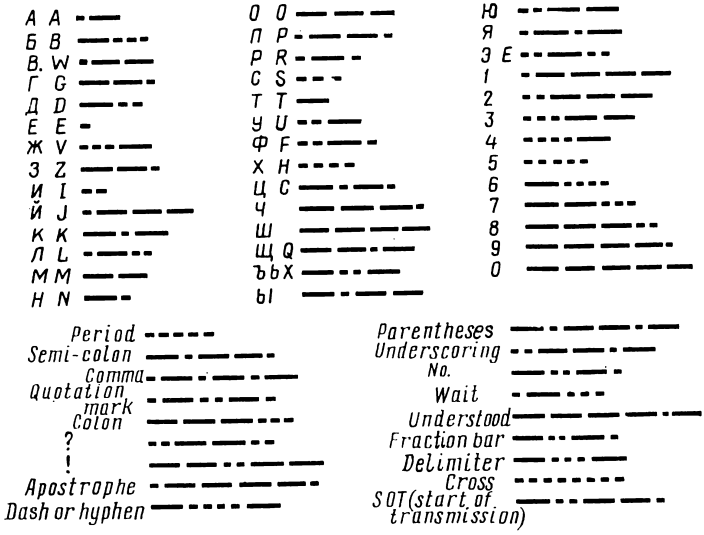

Fig. 3.1. Morse code

characters are assigned to the punctuation marks and special functions which usually occur more seldom.

The Morse code is a nonuniform-length code. Among its advantages are the ease with which it can be memorized, simple apparatus required to print it, the possibility for reception visually and aurally, and a relatively small total number of symbols used for the transmission of texts. This is why the Morse code has won world-wide recognition.

A major disadvantage of the Morse code, which limits its use, is the need to decode messages before they can be forwarded to the addresses. This is why considerable effort has been put in to develop automatic decoders, that is, letter-printing telegraph equipment. However, the Morse code has proved a poor match to an automatic decoder. Instead, use is now being made of equal-

length codes such as the simple binary five-unit Baudot code (Fig. 3.2).

Like Morse, Baudot arranged the characters of the alphabet in the order of their frequency in texts. The most commonly used letters were placed at the head of the alphabet, and the whole alphabet was divided into four parts, *I*, *II*, *III* and *IV* (seven characters each). Within each part the letters were coded into three-digit numerals *100*, *010*, *001*, *110*, *101*, *011*, and *111*, and the No. of a part as a two-digit numeral, *00*, *10*, *01* or *11*. Each letter of a text needed a five-unit code combination made up of a symbol code and a part code. Accordingly, the Baudot transmitter had five keys, three operated by the operator's right hand and two by his left hand. At the rate set by the distributor, the operator had to press the keys if the combination contained the unity code. In the Baudot code, the most frequently used letters were assigned the minimum number of unit elements and could be transmitted by the right hand alone. As is seen from Fig. 3.2, numerals and special functions were coded like letters. To identify them, separate "figures" and "letters" characters were included.

	Roman		Russian		Right hand			Left hand	
	1	A	1	A	1	0	0	0	0
	2	E	2	E	0	1	0	0	0
	3	T	3	И	0	0	1	0	0
I	&	E	ь	Я	1	1	0	0	0
	4	U	4	У	1	0	1	0	0
		L	"		0	1	1	0	0
	5	O	5	O	1	1	1	0	0
	6	J	6	Ш	1	0	0	1	0
	7	G	7	Г	0	1	0	1	0
II	8	B	8	Б	0	0	1	1	0
		H	!	Х	1	1	0	1	0
	9	C	9	Ц	1	0	1	1	0
	+	F	+	Ф	0	1	1	1	0
	0	D	0	Д	1	1	1	1	0
		.	.	Ч	1	0	0	0	1
	)	X	)	З	0	1	0	0	1
	;	S	;	С	0	0	1	0	1
III	:	Z	:	Ж	1	1	0	0	1
		T	Э	Т	1	0	1	0	1
	?	W	?	В	0	1	1	0	1
	,	V	Й	Ю	1	1	1	0	1
	(	K	Щ	К	1	0	0	1	1
	)	M	"	М	0	1	0	1	1
	-	R	-	Р	0	0	1	1	1
IV	=	L	=	Л	1	1	0	1	1
	/	Q	/	Ы	1	0	1	1	1
	N°	N	N°	Н	0	1	1	1	1
	%	P	%	П	1	1	1	1	1
	Figures				0	0	0	1	0
	Letters				0	0	0	0	1
	*	*	*	*	0	0	0	1	1

Fig. 3.2. Baudot code

About a quarter of a century after Baudot had invented his transmitter, Murrey invented a keyboard telegraph transmitter with a keyboard similar to that of an ordinary typewriter, which greatly simplified operation.

In the early 20th century, several telegraph codes were in use, which impeded international telegraph communication. Therefore, the International Telegraph Consultative Committee (CCITT in French) in 1931 adopted standard international code No. 1 which did not practically differ from the Baudot code widely used for page printers.

In 1932, standard international code No. 2 (ITC-2) was adopted for start-stop keyboard telegraph systems (Fig. 3.3). It has 32 information and function symbols in a binary five-unit code.

In the Soviet Union, use is made of a modified No. 2 international code (designated RVMK-2) adapted to the syllabary and alphabet of the Russian language and to working over international lines by means of type STA start-stop teleprinters. A type STA

Combination No.	Case			Case		Combination					Combination No.	Case			Case		Combination				
	Roman	Russian	Figs	Ltrs	Figs	1	2	3	4	5		Roman	Russian	Figs	Ltrs	Figs	1	2	3	4	5
1	A	А	–	A	–	1	1	0	0	0	17	Q	Я	1	Q	1	1	1	1	0	1
2	B	Б	?	B	?	1	0	0	1	1	18	R	Р	4	R	4	0	1	0	1	0
3	C	Ц	:	C	:	0	1	1	1	0	19	S	С	a'n	S	a'n	1	0	1	0	0
4	D	Д	Who are you?	D	$	1	0	0	1	0	20	T	Т	5	T	5	0	0	0	0	1
5	E	Е	3	E	3	1	0	0	0	0	21	U	У	7	U	7	1	1	1	0	0
6	F	Ф	Э	F	!	1	0	1	1	0	22	V	Ж	=	V	=	0	1	1	1	1
7	G	Г	Ш	G	&	0	1	0	1	1	23	W	В	2	W	2	1	1	0	0	1
8	H	Х	Щ	H	#	0	0	1	0	1	24	X	Ь	/	X	/	1	0	1	1	1
9	I	И	8	I	8	0	1	1	0	0	25	Y	Ы	6	Y	6	1	0	1	0	1
10	J	Й	Ю	У	Ω	1	1	0	1	0	26	Z	З	+	Z	+	1	0	0	0	1
11	K	К	(	K	(	1	1	1	1	0	27	Carriage return			Carriage return		0	0	0	1	0
12	L	Л	)	L	)	0	1	0	0	1	28	Line feed			Line feed		0	1	0	0	0
13	M	М	·	M	·	0	0	1	1	1	29	Roman			Ltrs		1	1	1	1	1
14	N	Н	,	N	1	0	0	1	1	0	30	Figs			Figs		1	1	0	1	1
15	O	О	9	O	9	0	0	0	1	1	31	Word space			Word space		0	0	1	0	0
16	P	П	0	P	0	0	1	1	0	1	32	Russian			Not used		0	0	0	0	0

Fig. 3.3. International standard code No. 2

teleprinter has three cases labelled ROMAN (letters), RUSSIAN (letters) and FIGURES. In the ROMAN and FIGURES cases, it is the same as international code No. 2. Transition to the Russian letters is indicated by a combination of five noughts. Since the 31 characters of the Russian alphabet cannot be arranged in the same order as the Roman letters, five Russian letters have been placed in the FIGURES case. Almost unmodified, standard international code No. 2 is used in type RTA-50-2 and STA-2M teleprinters.

3.2. CODES USED FOR ENTRY OF DATA TO DIGITAL COMPUTERS

From a look at the history of computers, it will be seen that at first they were used to do arithmetic, and the input information was numbers. If the numbers were presented in decimal notation, but a computer could only handle binary numbers, the need would arise for the conversion of decimal to binary numbers and back.

First-generation computers mainly used five-unit international code No. 2. As computers developed, it became necessary to add more function, arithmetic, logical and other symbols not provided by international code No. 2. As a result, several modifications of this code appeared. However, advances in computers and telegraphy showed that the five-unit code could not satisfy the growing needs, and a variety of six- and seven-unit codes were proposed.

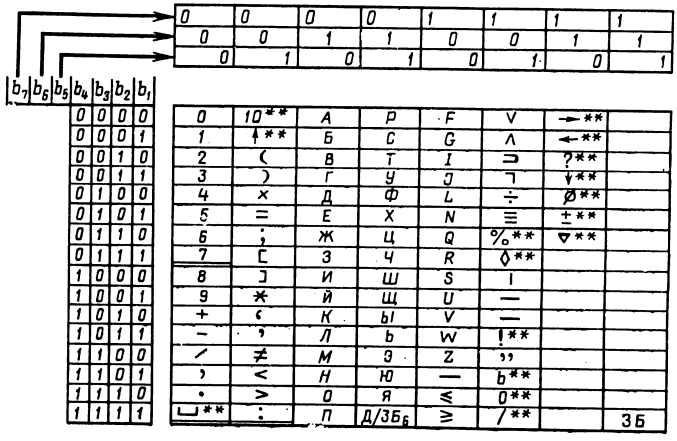

Fig. 3.4. Seven-unit code to GOST 10859-64

Under USSR State Standard GOST 10859-64 enacted in 1964, a seven-unit code is used to move data into and out of a computer (Fig. 3.4). The code table is so arranged that a subset of combinations using fewer units per character may be derived by simply deleting the most significant positions, if necessary. For example, the numerals 0 through 7 may be coded as three-unit binary combinations, all symbols in the first column including all decimal numerals, the "+" and "—" signs, the slash, the coma, the point, and space. The addition of the fifth unit enables 16 more mathematical symbols and special functions to be encoded. Notably, the upward arrow and the square brackets make it possible to print in line all power functions (because $a \uparrow x$ stands for a to power x), and all subscripted quantities (a [i] designates "a subscripted by i").

A six-unit code can represent all numerals and the letters of the Russian alphabet, while the complete seven-unit code additionally

includes some Roman letters (different in shape from the Russian ones), symbols for relations and logic functions, mathematical and special functions, and several reserve positions. The total number of reserve positions is thus now twenty-four, and they may be used for symbols which may appear in the future. The new seven-unit international primary code (Fig. 3.5) is built along about the same lines (Fig. 3.5). Logically, it may be divided into four zones or columns: *0 — 1*, *2 — 3*, *4 — 5*, and *6 — 7*.

b_7	b_6	b_5	b_4	b_3	b_2	b_1		0	0	0	0	1	1	1	1
								0	0	1	1	0	0	1	1
								0	1	0	1	0	1	0	1
								0	1	2	3	4	5	6	7
0	0	0	0				0	NUL	(TC_7)DLE	Space	0	@	P	\	p
0	0	0	1				1	(TC_1)SOH	DC_1	!	1	A	Q	a	q
0	0	1	0				2	(TC_2)STX	DC_2	"	2	B	R	b	r
0	0	1	1				3	(TC_3)ETX	DC_3	# £	3	C	S	c	s
0	1	0	0				4	(TC_4)EOT	DC_4	$	4	D	T	d	t
0	1	0	1				5	(TC_5)ENQ	(TC_8)NACK	%	5	E	U	e	u
0	1	1	0				6	(TC_6)ACK	(TC_9)SYNC	&	6	F	V	f	v
0	1	1	1				7	BEL	(TC_{10})ETB	'	7	G	W	g	w
1	0	0	0				8	(FE_0)BS	CAN	(	8	H	X	h	x
1	0	0	1				9	(FE_1)HT	EM	)	9	I	Y	i	y
1	0	1	0				10	(FE_2)LF	SUB	*	:	J	Z	j	z
1	0	1	1				11	(FE_3)VT	ESC	+	;	K	(Ä)	k	
1	1	0	0				12	(FE_4)FF	(IS_4)FS	,	<	L		l	
1	1	0	1				13	(FE_5)CR	(IS_3)GS	-	=	M	(Ü)	m	
1	1	1	0				14	SO	(IS_2)RS	.	>	N	^	n	
1	1	1	1				15	SI	(IS_1)US	/	?	O	-	o	DEL

Fig. 3.5. ISO seven-bit code

The first zone contains functional symbols, except DEL (delete) for which the last position is reserved (the combination 1111111). The core of the functional symbols in this zone is divided into four groups as follows:

group one, to control transfer of information over communication channels (designated $TC_1 — TC_{10}$ in the table);

group two, to control print-out (designated $FE_4 — FE_5$);

group three, to control terminal devices (designated DC_1 through DC_4);

group four, to separate information (designated $IS_1 — IS_4$).

The remaining zones of the code table are occupied by printable (graphic) symbols. Columns *2-3* are allocated to the special mathematical symbols, punctuation marks and numerals.

The last two zones contain upper- and lower-case Roman letters arranged in accordance with appropriate lexical and graphical

rules (the positions of the numerals, punctuation marks and space are likewise chosen to satisfy these requirements). All reserve positions are located here, too.

The SPACE symbol is not usually printed, yet it is classed as printable and is used to separate words and advance the text one position forward.

As is seen from the table, the numerals are encoded in the ordinary binary code. Four-unit combinations are constructed for them by discarding the three most significant units.

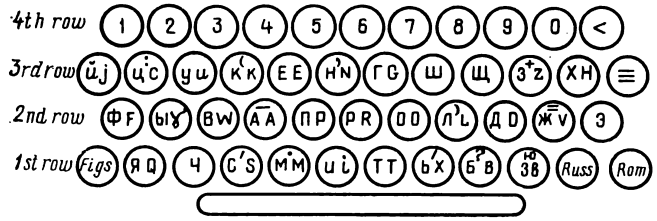

Fig. 3.6. Four-row keyboard

Soviet teleprinters have a four-row keyboard (Fig. 3.6) so they look like typewriters.

The new international code has been developed for texts written in alphabets using Roman letters. Because of this it cannot be used in the USSR. As a way out, a new modification of the primary standard code (GOST 13052-67) was developed in 1967 (Fig. 3.7). In this code, the "0" symbol designates the Roman case, and the "1" symbol, the Russian case. The Russian case has the same zones as the international one.

The symbols in the second zone are left unchanged. The first currency sign is represented by a symbol recommended by the third code draft, and the second currency symbol is omitted. The 2/3 position is used for the "No" symbol. The last two zones are occupied completely (except the DELETE and UNDERSCORE positions) by Russian upper- and lower-case letters. The fact that the lower-case letters are placed ahead of the upper-case ones reflects the desire to use (where lower-case letters are not essential) a code with only upper-case letters, and do so with the least expense on code conversion.

In the ES EVM computers (the Russian abbreviation for "a unified system of electronic computers"), information is put in and

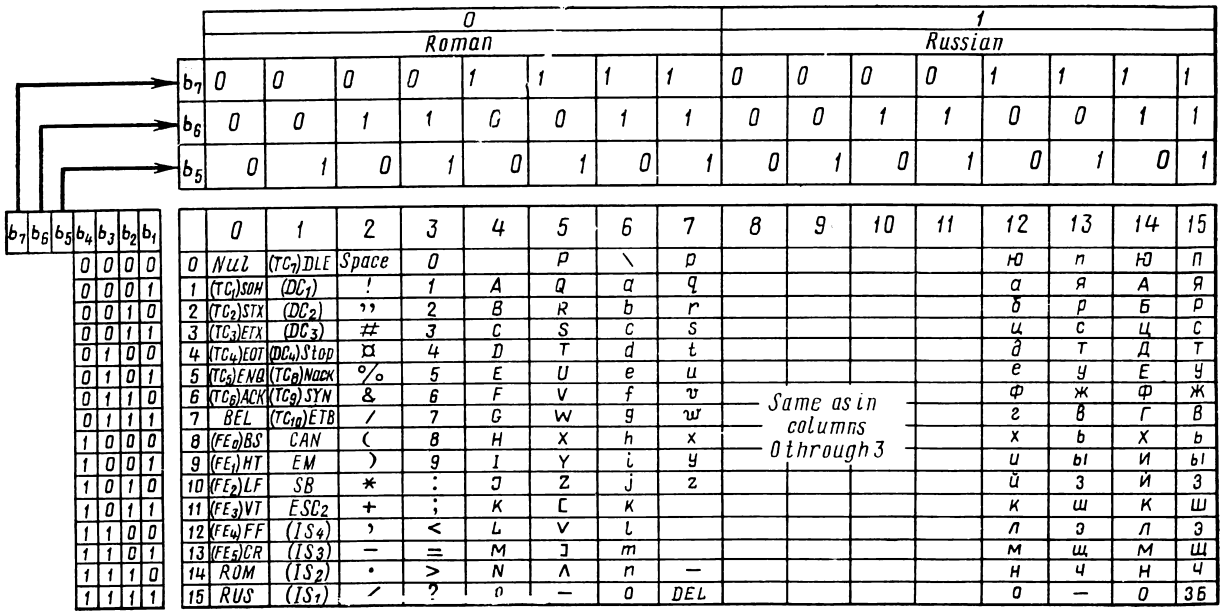
0
Roman
1
Russian
b7 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
b6 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
b5 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
b7 b6 b5 b4 b3 b2 b1
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 NUL (TC7)DLE Space 0 P \ p
1 (TC1)SOH (DC1) ! 1 A Q a q
2 (TC2)STX (DC2) " 2 B R b r
3 (TC3)ETX (DC3) # 3 C S c s
4 (TC4)EOT (DC4)Stop ¤ 4 D T d t
5 (TC5)ENQ (TC8)NACK % 5 E U e u
6 (TC6)ACK (TC9)SYN & 6 F V f v
7 BEL (TC10)ETB ' 7 G W g w
8 (FE0)BS CAN (8 H X h x
9 (FE1)HT EM) 9 I Y i y
10 (FE2)LF SB * : J Z j z
11 (FE3)VT ESC2 + ; K [k
12 (FE4)FF (IS4) , < L V l
13 (FE5)CR (IS3) - = M] m
14 ROM (IS2) . > N ^ n —
15 RUS (IS1) / ? O _ o DEL
Same as in columns 0 through 3
б ю п ю п
а я А я
б р Б р
ц с Ц с
д т Д т
е у Е у
ф ж Ф ж
г в Г в
х ь Х ь
и ы И ы
й з И з
к ш К ш
л э Л э
м щ М щ
н ч Н ч
о — О Зб

Fig. 3.7. Standard code to GOST 13052-67

out of a computer by use of the KOI-8 information interchange code (Fig. 3.8). This code is based on the eight-unit code (to GOST

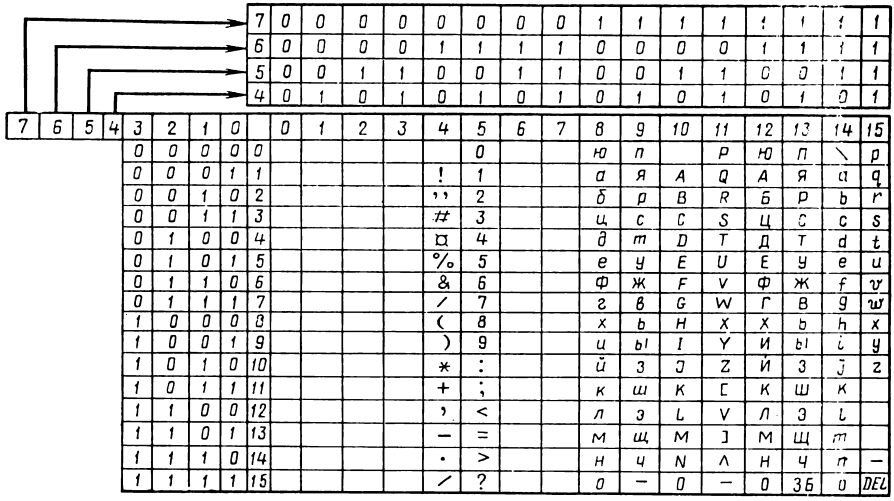

Fig. 3.8. KOI-8 (eight-bit information interchange) code

13052-67), but has a somewhat modified list of functional symbols and a modified arrangement of the columns containing other symbols.

3.3. ERROR-DETECTING AND ERROR-CORRECTING CODES

The term "error-detecting and/or error-correcting" applies t codes which incorporate features intended to detect and corr errors that may occur in transmission due to noise. Such fea usually take the form of some redundance.

According to the manner in which this redundance is introduced and utilized, error-detecting and/or error-correcting codes may be separable and inseparable.

In an inseparable code, message symbols are not clearly separated from check symbols. In a separable code, they are clearly separated, and each n-unit code combination contains k message symbols ($k < n$) and $n - k$ check symbols. This arrangement enables any primary code to be endowed with an error-detecting and/or error-correcting capacity.

A simple form of this redundance is the parity check. To this end, a check bit which is a modulo 2 sum of all message bits is added to each character. The sum of all message bits in a character may be 0 or 1, that is, be equal to an even or odd number of ones. In an error-correcting code incorporating a parity-check bit, the overall sum of ones is made always to be even (or odd) by placing a 0 or 1 in the check-bit position. As data are read into a computer, each character is checked for parity. If the check shows the correct sum, the character is taken to have been entered correctly. This form of check can reveal single errors (or any odd number of single errors), and fails where an even number of errors occurs. An advantage of single error-detecting codes utilizing the parity check is simplicity; a major limitation is that only single errors can be detected and nothing is provided by way of correcting them.

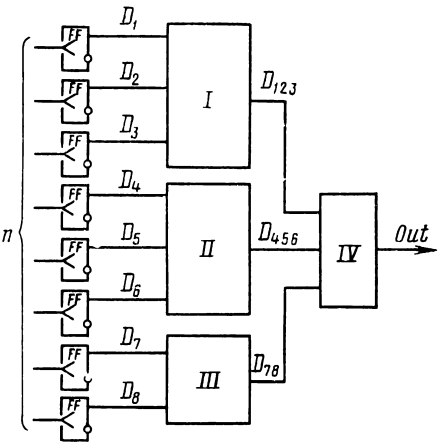

Fig. 3.9. Odd-parity check scheme

A practical realization of an odd parity check is illustrated in Fig. 3.9. The circuit uses threshold-logic matrices. The eight-digit number (containing seven information digits and one check digit) written into the register is transferred to the matrices. The first three places are entered in matrix *I*, the fourth, fifth and sixth in matrix *II*, and the last two places in matrix *III*, where the following relations are established:

matrix I

D_1	D_2	D_3	D_{123}
0	0	0	1
1	0	0	0
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	1
0	1	1	1
1	1	1	0

matrix II

D_4	D_5	D_6	D_{456}
0	0	0	1
1	0	0	0
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	1
0	1	1	1
1	1	1	0

matrix III

D_7	D_8	D_{78}
0	0	1
1	0	0
0	1	0
1	1	1

From the outputs of matrices *I*, *II* and *III*, the signals D_{123}, D_{456} and D_{78} are applied to matrix *IV* which instruments the following relations:

matrix IV

D_{123}	D_{456}	D_{78}	*OUT*
0	0	0	0
1	0	0	1
0	1	0	1
1	1	0	0
0	0	1	1
1	0	1	0
0	1	1	0
1	1	1	1

If the signal at the output of matrix *IV* be 0, no clock pulse will be allowed to pass on to other circuits; if the output signal be 1, a clock will pass on.

Both separable and inseparable codes may be of the *block* and the *nonblock* type. In a block code, each information sequence is divided into blocks of a definite length and check digits are added at the end of each block; hence all checking is done likewise on blocks of definite length. In a nonblock code, information sequences are not divided into fixed-length blocks, and check symbols are interspersed between information symbols to a specific algorithm.

If all code combinations of a block code have the property of symmetry, that is, if any code combination is equally distant from other combinations, this is a *group* code.

A further division of block codes is into *systematic* and *nonsystematic*. With a systematic code, linear operation (addition modulo 2) on any combinations yields a combination of the same code. Among the systematic codes, the most important for practical use are cyclic codes, that is, codes in which the combinations have the cyclic property when their elements are permutated consecutively. Nonsystematic codes are formed by performing nonlinear operation on information symbols.

A complete set of nonzero combinations of an *equal-length n-unit code* may be defined by an *identity* matrix.

An *identity matrix* $\mathbf{J}_n$ is an $n \times n$ matrix, that is one with n rows and n columns, which has 1's on the main or leading diagonal,

and 0's elsewhere

$$\mathbf{J}_n = \begin{Vmatrix} 10 \ldots 000 \\ 01 \ldots 000 \\ \cdots\cdots \\ 00 \ldots 010 \\ 00 \ldots 001 \end{Vmatrix} \tag{3.1}$$

All rows (columns) of the identity matrix are linearly independent because the sum of any combinations is nonzero. A unit matrix of the form (3.1) is a defining one for an equal-length code because sequential modulo 2 addition of its rows in all likely combinations can yield $N = 2^n - 1$ nonzero code combinations. For example, for a three-bit code

$$\mathbf{J}_n = \begin{Vmatrix} 100 \\ 010 \\ 001 \end{Vmatrix}; \qquad N = 2^n - 1 = 7$$

and these combinations are 100, 010, 001, 011, 101, 110 and 111.

During the transmission of an n-bit code combination, an error may arise because some of the transmitted symbols are replaced by incorrect ones (say, a 1 by a 0 or vice versa).

As a way of detecting errors, instead of all $N_0 = 2^n$ possible combinations, an equal-length code uses for transmission only some allowed combinations:

$$N_0 = 2^k < 2^n \tag{3.2}$$

The remaining $2^n - 2^k$ combinations are called *forbidden* and are used for error detection. Whenever a forbidden combination appears in the received signal, this is an indication of an error, because it can only happen when some allowed combination has been mutilated. Should, however, mutilation of some allowed code combination result in another allowed combination, the error would be passed unnoticed. The mechanism by which a binary code combination is mutilated, that is, an allowed combination is reduced to a forbidden combination $\mathbf{B}_{it}$, may be visualized as addition modulo 2 of code combination vectors $\mathbf{A}_i$ and error vectors $\mathbf{e}_t$:

$$\mathbf{B}_{it} = \mathbf{A}_i \oplus \mathbf{e}_t \tag{3.3}$$

where the "$\oplus$" sign means addition modulo 2. The vectors $\mathbf{e}_t$ represent all likely combinations of errors; the number of 1's in them is equal to error multiplicity, t.

With this arrangement, the relative number of detected errors is

$$\frac{2^k(2^n - 2^k)}{2^k \times 2^n} = 1 - (1/2^{n-k}) \tag{3.4}$$

where $2^k(2^n - 2^k)$ is the number of transitions to forbidden combinations that can be detected, and $2^k \times 2^n$ is the number of likely transitions of allowed combinations into any other, including allowed ones.

So that errors can be not only detected but also corrected, the set of forbidden combinations, $\{B_j\}$, where $j = 1, 2, \ldots, 2^n - 2^k$, is decomposed into N non-intersecting subsets, M_i, where $i = 1, 2, \ldots, N$. Each of the subsets M_i is assigned to one of the allowed code combinations A_i, thereby establishing their relation to one of the subsets of forbidden combinations M_i. Therefore, each time any forbidden combination out of this subset M_i appears, the error can be corrected by changing to the assigned allowed combination A_i. The number of erratic combinations that can be corrected is $2^n - 2^k$.

The fractional number of erratic combinations that can be corrected is

$$\frac{2^n - 2^k}{2^k(2^n - 2^k)} 1/2^k \tag{3.5}$$

The manner in which the set is to be decomposed into subsets M_i must be correlated with the error statistics.

As an example, suppose use is made of the following random combinations ($k = 2$): $A_1 = 0001$, $A_2 = 0101$, $A_3 = 1110$, and $A_4 = 1111$. Let these combinations be subjected to errors (all likely erratic combinations B_{it} are summarized in Table 3.1).

It is seen from Table 3.1 that errors varying in multiplicity give rise to forbidden combinations which may be the same in different columns. These combinations must be deleted so as to preserve one-to-one correspondence between M_i and A_i. For this purpose, we shall cut down the multiplicity of corrected errors by crossing out any error-produced combinations starting with $t = 3$ and $t = 4$. However, the remaining forbidden combinations still include a proportion of identical ones. Therefore, we inspect the table again, starting from the left end of the top row and delete all repeating combinations. As a result, we obtain a table like 3.2, which contains only non-recurring forbidden combinations. The combinations of each column are combined into a subset M_i.

Table 3.1

e_t	A_i				t
	A_1	A_2	A_3	A_4	
0001	0000	0100	1111 *	1110 *	1
0010	0011	0111	1100	1101	
0100	0101 *	0001 *	1010	1011	
1000	1001	1101	0110	0111	
0011	0010	0110	1101	1100	2
0101	0100	0000	1011	1010	
1001	1000	1100	0111	0110	
0110	0111	0011	1000	1001	
1010	1011	1111 *	0100	0101 *	
1100	1101	1001	0010	0011	
0111	0110	0010	1001	1000	3
1011	1010	1110 *	0101 *	0100	
1101	1100	1000	0011	0010	
1110	1111 *	1011	0000	0001 *	
1111	1110 *	1010	0001 *	0000	4

* Mutilated combinations identical with any allowed combination cannot be recognized as mutilated.

Table 3.2

A_i				t
A_1	A_2	A_3	A_4	
0000	0100	1100	1101	1
0011	0111	1010	1011	
1001	—	0110	—	
0010	—	1000	—	2
M_1	M_2	M_3	M_4	—

The remaining table is seen to contain ten single-error combinations and two two-error combinations. The error-correcting code developed in this way can correct a multiplicity of single errors and two double errors, that is to recover the correct A_i, as soon as the remaining forbidden combinations turn up in the transmitted message. For example, the forbidden combination 0111 in subset M_2 permits the reconstruction of the allowed combination A_2 (0101) assigned to this subset.

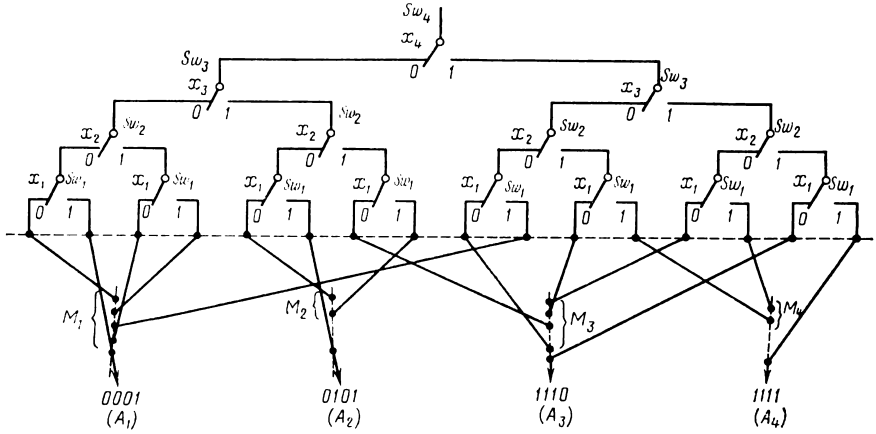

Fig. 3.10. Decoding scheme

A similar table of forbidden combinations might be compiled for a higher multiplicity of errors. In the general case, however, it is advisable to develop an error-correcting code in knowledge of error statistics, so that the most probable ones will be corrected.

An error-correcting code is implemented by the operation of decoding which, in the general case, reduces to comparing the received combinations with all the 2^n likely combinations. If a received code combination checks with any one of the allowed combinations, it is classed as legitimate; if it is identified with any forbidden combination out of the subset M_i, the correct combination is reconstructed. A decoding scheme for a code having $n = 4$ and $k = 2$ is shown in Fig. 3.10. It consists of two-position switches (relays) keyed by code signals of the respective bit positions. A 1 signal causes the respective switch to close its right-hand contact, while a 0 signal causes it to close its left-hand contact. The position of the switch Sw_4 is controlled by the code

signal associated with the most significant bit x_4 in the received combination; that of Sw_3 by the signal of the next less significant bit x_3, and so on. Depending on the signals and, as a consequence, the positions of the switches, the received combination may appear either on any one of the four output wires assigned to legitimate (correct) combinations, or on any one of the twelve wires allocated to forbidden combinations. The forbidden-combination wires are grouped in accordance with subsets M_i. Once a mutilated combination out of the subset M_i appears, it is corrected, and presented at the output as the correct combination A_i.

Practical utilization of an error-correcting code depends to a considerable degree on how simple the decoding operation can be instrumented.

In an n-unit binary code where all likely combinations of binary digits are used as legitimate combinations, the Hamming distance * is $d = 1$, and any single error alters one of the legitimate combinations into another. Therefore, no error can be detected. However, this is not to be construed that if $d = 1$ the code has no corrective ability. The idea will be best understood from the following example using a four-unit code (see Fig. 3.10) where the following legitimate combinations are utilized: 0001, 0101, 1110 and 1111. For this code, d defined as the minimum distance between combinations (A_1 and A_2, A_3 and A_4) is unity. Still, the code will fail to detect only single errors causing the transitions $A_1 \leftrightarrows A_2$ and $A_3 \rightleftarrows A_4$, while any other single errors will be detected. For all errors without exception to be detected, it is essential that $d \geqslant 2$.

If we apply this reasoning to errors of multiplicity t_0, we may conclude that for them to be detected it is vital that

$$d \geqslant t_0 + 1 \tag{3.6}$$

Thus, in order to detect all single errors, legitimate combinations ought not to be selected by chance; rather, this should be done so as to obtain the longest Hamming distance. For example, code combinations for which $d = 2$ may be 0001, 0111, 1110 and 1011.

Correctable errors of multiplicity t_c are connected to the Hamming distance by the following relation

$$d = 2t_c + 1 \tag{3.7}$$

---

* The number of positions in which the code words differ. — Tr.

An important feature of any error-correcting code is redundancy. This may be absolute and relative.

If each n-digit code combination has k information digits and $n - k$ check digits, the *absolute redundancy* will be

$$U = n - k \tag{3.8}$$

The *relative redundancy* will then be defined as

$$U_r = (n - k)/n \tag{3.9}$$

or

$$U_r = (n - k)/k \tag{3.10}$$

Codes securing a desired corrective capability at minimum redundancy are called *optimal*. To state this differently, for a given error distribution in a communication channel an optimal code should give the least probability of incorrect reception of code combinations.

At present, a general method for developing optimal linear codes is non-existent. We only have some bounds on the functional relation

$$n - k = f(k, d) \tag{3.11}$$

In approximate terms, the necessary redundancy of a code for a specified distance d may be found from the upper bound on $n - k$, called the *Hamming bound*, which is derived as follows.

All forbidden code combinations likely to appear due to errors of multiplicity from 1 to t_n are grouped into subsets M_i. The number of combinations in a subset M_i is $\sum_{t=1}^{t_n} C_n^t$, where C_n^t is the number of groups of n elements taken t at a time; it defines how many errors of multiplicity t may occur within the length of an n-unit code combination. Adding the legitimate code combination A_i assigned to the subset M_i, we finally get $1 + \sum_{t=1}^{t_n} C_n^t$.

Recalling that the number of legitimate combinations (and, as a consequence, that of subsets M_i) is 2^k, the total number of combinations in non-intersecting subsets is

$$2^k \left(1 + \sum_{t=1}^{t_n} C_n^t\right) \tag{3.12}$$

Since the maximum number of distinguishable code combinations is 2^n, we get

$$2^k\left(1+\sum_{t=1}^{t_n} C_n^t\right)\leqslant 2^n$$

whence

$$2^{n-k}\geqslant 1+\sum_{t=1}^{t_n} C_n^t \tag{3.13}$$

or

$$n-k\geqslant \log_2\left(1+\sum_{t=1}^{t_n} C_n^t\right) \tag{3.14}$$

Re-writing (3.14) gives

$$n-k\geqslant \log_2\sum_{t=0}^{t_n} C_n^t \tag{3.15}$$

For example, if $n=7$ and $t_n=1$, then, by (3.7), $d=3$, and $n-k\geqslant \log_2(1+7)=3$.

This bound holds only for binary codes correcting independent errors.

It is not always worth while using optimal or nearly optimal error-correcting codes. The likely advantages may be more than offset by the complexity of instrumenting coding and decoding schemes. From reference to the circuit of Fig. 3.10, it is seen, for example, that such a decoder would require 2^n-1 switches, and their number would grow with increasing code length. Therefore, such an arrangement is only warranted in the case of short-length codes. This limitation has prompted the search for simpler decoding schemes.

Decoding techniques simpler than direct comparison of a received code combination with all likely ones are offered by the rules embodied in the structure of the codes examined below.

3.4. SYSTEMATIC CODES

The name '*systematic*' applies to any group n-unit code in which out of the n symbols making up a code combination, k symbols are information (or intelligence) units and the remaining $n-k$ units are redundant, intended solely for checking purposes (these check bits take up the same positions in all combinations).

A systematic code is formed by a generator and a parity-check matrix. The *generator matrix* generates all likely code combinations by taking modulo 2 sums of all likely combinations of rows (two, three, etc., taken at a time). The generator matrix provides a basis on which to build the *parity-check matrix* which is utilized to develop coding and decoding algorithms, that is, to synthesize a coder and a decoder.

The generator matrix is developed by writing an additional matrix containing $n-k$ columns and k rows on the right of a $k \times k$ square identity matrix, for example

$$\mathbf{A} = \left\| \begin{array}{cc} \overbrace{10 \ldots 000}^{k} & \overbrace{a_{11}a_{12} \ldots a_1(n-k)}^{n-k} \\ 01 \ldots 000 & a_{21}a_{22} \ldots a_2(n-k) \\ \ldots\ldots\ldots & \ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots & \ldots\ldots\ldots\ldots \\ 00 \ldots 001 & a_{k1}a_{k2} \ldots a_k(n-k) \end{array} \right\|$$

The rows of the additional matrix are obtained by exhausting the various $n-k$ bit combinations containing at least $d-1$ ones such that the modulo 2 sum of any two rows should contain at least $d-2$ ones. This requirement stems from the fact that the Hamming (signal) distance depends on both the number of ones per row and their relative position along the code combination (each row of a unit matrix contains a unity, and the signal distance d between the combination matrix is equal to 2).

In developing the parity-check matrix, a matrix containing k columns and $n-k$ rows is pre-fixed to a $(n-k) \times (n-k)$ square identity matrix, with each row derived from the respective pre-fixed column of the generator matrix:

$$\mathbf{H} = \left\| \begin{array}{cc} a_{11}a_{21} \ldots a_{k1} & 10 \ldots\ldots 0 \\ a_{12}a_{22} \ldots a_{k2} & 01 \ldots\ldots 0 \\ \ldots\ldots\ldots\ldots\ldots & \ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots & \ldots\ldots\ldots \\ a_{1\,(n-k)}a_{2\,(n-k)} \ldots a_{k\,(n-k)} & 00 \ldots\ldots 1 \end{array} \right\|$$

From the parity-check matrix thus developed, it is possible to select a coding and decoding algorithm for a systematic code on the strength of the fact that the unities in each row correspond to the code characters whose modulo 2 sums are zero, that is, even.

As an example, consider a single-error correcting code ($t_c = 1$) having $n = 7$.

The signal distance is

$$d = 2t_c + 1 = 3$$

By (3.15), the number of check bits is

$$n - k = 3$$

and so the number of information bits must be

$$k = 4$$

The generator matrix takes the form

$$\mathbf{A} = \begin{Vmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{Vmatrix}$$

Now we take the modulo 2 sums of all possible combinations of rows and obtain the remaining code sequences. For example, the first and second rows give 1100110; the first and third, 1010101; the first and fourth, 1001100, etc; the first, second, third and fourth 1111111. Thus, the first four symbols represent all possible four-digit binary numbers.

Using the generator matrix, we may write the parity-check matrix as

$$\mathbf{H} = \begin{Vmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{Vmatrix}$$

From the parity-check matrix we may write the following parity-check rules for the first, second and third rows

$$(1)\quad r_1 = a_2 \oplus a_3 \oplus a_4 \oplus a_5 = 0$$

$$(2)\quad r_2 = a_1 \oplus a_3 \oplus a_4 \oplus a_6 = 0$$

$$(3)\quad r_3 = a_1 \oplus a_2 \oplus a_4 \oplus a_7 = 0$$

It is seen from the above parity-check rules that a_1 appears in Equations 2 and 3; a_2 in Equations 1 and 3; a_3 in Equations 1 and 2; and a_4 in all equations; while a_5, a_6, and a_7 appear each in only one equation. Thus, mutilation of any one of the elements a_i would affect certain definite equations from which the mutilated element can readily be detected and corrected. This procedure is known as the position parity check.

The check bits can be obtained from the parity-check rules for rows:

$$a_2 \oplus a_3 \oplus a_4 = a_5$$
$$a_1 \oplus a_3 \oplus a_4 = a_6$$
$$a_1 \oplus a_2 \oplus a_4 = a_7$$

If a source generates information sequencies in the above four-unit code a_1, a_2, a_3 and a_4, say 1100, the coder will generate the following sequence: a_1, a_2, a_3, a_4, a_5, a_6, a_7, that is, 1100110. This sequence, after transmission over a communication channel, is checked by a decoder. If the sequence has been mutilated on its way in the channel so that the sequence at the receiving end is 1101110, the decoder will show that

$$r_1 = a_2 \oplus a_3 \oplus a_4 \oplus a_5 = 1$$
$$r_2 = a_1 \oplus a_3 \oplus a_4 \oplus a_6 = 1$$
$$r_3 = a_1 \oplus a_2 \oplus a_4 \oplus a_7 = 1$$

that is, the mutilated elements is a_4, common to all equations. The correct combination can be restored by inverting the distorted bit.

Thus, the checking procedure yields a code sequence of the form $R_j = r_1 r_2, \ldots, r_{n-k}$ called the *parity-check sequence*. Each parity-check sequence may uniquely be associated with a correcting vector $\mathbf{E}_j$. When this vector is added modulo 2 to the received sequence $\mathbf{B}_j$, the resultant sequence will be the code sequence actually transmitted:

$$\mathbf{A}_j = \mathbf{B}_j \oplus \mathbf{E}_j \tag{3.16}$$

When $R_j = 0$, it is assumed that the code sequence has been received correctly or an undetectable error has occurred. Thus, the parity-check sequence assigned to each coset of legitimate code sequences makes it possible to identify any received sequence.

Obviously, the digit range of the $(n - k)$ parity-check sequence is decisive as regards the maximum number of errors that can be

detected or corrected, because it determines the maximum number of possible non-zero parity-check sequencies, which is equal to $2^{n-k} - 1$.

The parity-check sequence constructed in the manner described above signals the presence of an error, and its position can now be found through some additional steps.

Alternatively, the parity-check sequence can be so constructed that it will directly point to the bit in error; this will greatly simplify decoding. For this to be possible, it is necessary that the first check should yield the weight of the least significant place of the number, 2^0; the second check, the weight of the next more significant place, 2^1; etc. Then the sum of all checks will give the label of the erroneous place. Codes obeying this rule are called *Hamming codes* or *alphabets*.

Parity-check equations, or rules, for a Hamming code are derived as follows. All positions of a code sequence are labelled in binary form as follows: $a_1 = a_{0001}$, $a_2 = a_{0010}$, $a_3 = a_{0011}$, etc. The first parity-check rule is written as the modulo 2 sum of all places whose labels have a unity in their least significant position, 2^0:

$$r_1 = a_1 \oplus a_3 \oplus a_5 \oplus a_7$$

The second parity-check rule is written as the modulo 2 sum of all places whose labels have a unity in the second position, 2^1:

$$r_2 = a_2 \oplus a_3 \oplus a_6 \oplus a_7$$

The third parity-check rule is written as the modulo 2 sum of all places whose labels have a unity in the third position, 2^2, and so on.

Coding equations for check places are derived by equating the parity-check rules to zero in the absence of errors:

$$\left.\begin{aligned} a_1 &= a_3 \oplus a_5 \oplus a_7 \\ a_2 &= a_3 \oplus a_6 \oplus a_7 \\ &\cdots\cdots\cdots \\ &\text{etc.} \end{aligned}\right\} \qquad (3.17)$$

Within each code sequence, the check bits occupy places corresponding to their labels.

As an example, suppose we are to transmit the same information sequence, 1100, as in the previous example. There will be $n - k = 3$ check bits. In the Hamming code, a_1 will occupy the

first position, a_2 the second, and a_4 the fourth. The code sequence will take the form: $a_1a_2 1 a_4 100$.

The check rules yield the following results:

$$a_1 = a_3 \oplus a_5 \oplus a_7 = 1 \oplus 1 \oplus 0 = 0$$
$$a_2 = a_3 \oplus a_6 \oplus a_7 = 1 \oplus 0 \oplus 0 = 1$$
$$a_4 = a_5 \oplus a_6 \oplus a_7 = 1 \oplus 0 \oplus 0 = 1$$

Thus, the transmitted sequence containing the check places will be 0111100.

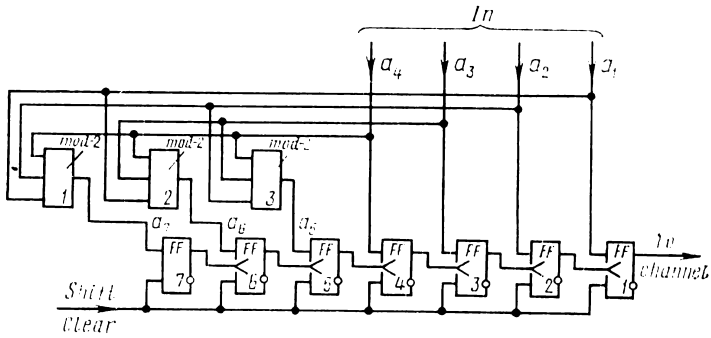

Fig. 3.11. Coder for a (7, 4) systematic code

Should a single error occur during transmission so that the received combination is 0101100, the decoder will yield the following result:

$$r_1 = a_1 \oplus a_3 \oplus a_5 \oplus a_7 = 0 \oplus 0 \oplus 1 \oplus 0 = 1$$
$$r_2 = a_2 \oplus a_3 \oplus a_6 \oplus a_7 = 1 \oplus 0 \oplus 0 \oplus 0 = 1$$
$$r_3 = a_4 \oplus a_5 \oplus a_6 \oplus a_7 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

that is, $r_1r_2r_3 \rightarrow 1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 = 3$.

That is, the mutilated bit is in the third position, and the 0 in that position should be altered to a 1.

By the rules of systematic codes set forth above, the coding and decoding of information should proceed as follows.

The coder should contain an n-bit register and $r = (n - k)$ modulo 2 adders, each intended to generate one check bit from certain information bits to an algorithm specified by the generator matrix.

Using the previous example, consider operation of a coder such as shown in logic-diagram form in Fig. 3.11. Prior to operation,

all flip-flops of the coding register are set to zero. The four-unit sequence to be coded is applied to the input in parallel form and fills the first four flip-flops of the coding register. At the same time, these information bits are applied to the modulo 2 adders in an order determined by the coding equations. The modulo 2 sums fill then the remaining three flip-flops of the register. Now the operation of coding is complete. Shift pulses cause the coded sequence to be moved to a transmitter. When the next sequence comes n clock periods later, the cycle is repeated all over again.

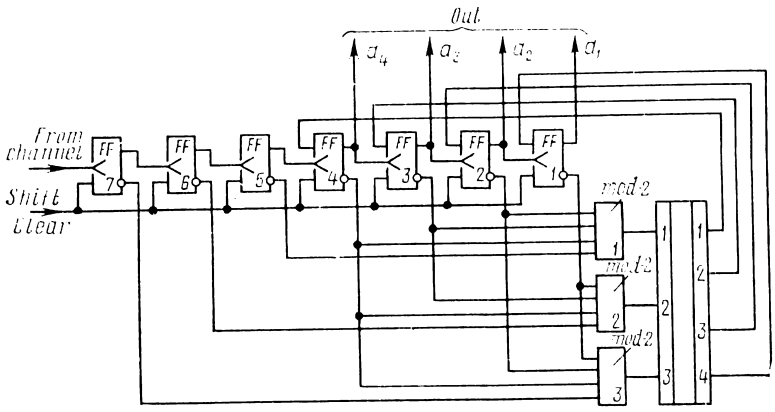

Fig. 3.12. Decoder for a (7, 4) systematic code

In the case of the Hamming code, the coder is basically the same, except that the inputs and outputs of the adders and the inputs of the flip-flops intended to store information bits are connected in a somewhat different way.

A decoder should contain an n-bit shift register, $r = (n - k)$ modulo 2 adders, and a parity-check sequence decoder to generate the correcting vector. In logic-diagram form, an error-correcting decoder for the previous example is shown in Fig. 3.12. Consider its operation.

The data item coming from a communication channel sequentially fills the seven-bit decoding register. After n clock periods, the modulo 2 adders will have formed a parity-check sequence (the adder inputs are coupled to the outputs of the flip-flops used in the decoding register by the check rules). The parity-check sequence is applied to the parity-check sequence decoder

which generates a correcting vector (the decoder has $2^r - 1$ outputs corresponding to a particular parity-check sequence). The correcting vector is applied to the set-reset inputs of the flip-flops in the information positions of the register, and is added during the $(n + 1)$st clock period with the erroneous code sequence; this results in the correction of the most probable error. The operation is carried out by simply inverting the state of the flip-flop. The corrected sequence is put out of the register and appears at the output. If the decoder is solely intended to detect errors, the outputs of all modulo 2 adders are tied together to generate an error-detecting signal. The Hamming code is decoded in a similar manner.

In the general case, systematic codes need simpler coders and decoders.

3.5. CYCLIC CODES

Attempts to simplify the coding and decoding of systematic codes have resulted in the appearance of codes called cyclic. The reason for naming them so is that allowed code sequences are formed as a result of cyclic permutations. For example, the cyclic permutation of the sequence $a_0a_1a_2 \ldots a_{n-1}$ will yield the sequence $a_{n-1}a_0a_1 \ldots a_{n-2}$.

In dealing with cyclic codes, it is convenient to re-cast code sequencies into polynomials of order $n - 1$:

$$a_0 + a_1x^1 + a_2x^2 + \ldots + a_{n-1}x^{n-1} \tag{3.18}$$

so that all operations on the code sequences reduce to those on the polynomials. Without going any further into the theory of cyclic codes, it may be noted that cyclic permutation boils down to the simple multiplication of the polynomial by x; if this yields x^n, it is replaced by unity, and the latter is placed instead of a zero. For example, let there be a code sequence, 1101011. It may be written as a polynomial of the form

$$1 + x^1 + x^3 + x^5 + x^6$$

To carry out cyclic permutation, we multiply this polynomial by x, and this will give

$$x^1 + x^2 + x^4 + x^6 + x^7$$

After we replace x^7 by 1 and place it in the position held by zero, we get 1110101.

The sum modulo 2 of the original sequence and the one formed by cyclic shift

$$\begin{array}{r} 1101011 \\ \oplus\ 1110101 \\ \hline 0011110 \end{array}$$

can be obtained by multiplying together the polynomial of the original sequence and the binomial $(1 + x)$:

$$\begin{array}{l} 1 + x + 0 + x^3 + 0 + x^5 + x^6 \\ \qquad\qquad\qquad\qquad\qquad\qquad 1 + x \\ \hline x + x^2 + 0 + x^4 + 0 + x^6 + x^7 \\ 1 + x + 0 + x^3 + 0 + x^5 + x^6 \\ \hline 1 + 0 + x^2 + x^3 + x^4 + x^5 + 0 + x^7 \\ \uparrow\ldots\ldots\ldots\ldots\ldots\ldots\ldots\mid \\ \hline \end{array}$$

This polynomial represents the code sequence 0011110.

A cyclic code may be specified in terms of a generator polynomial, $g(x)$ of order $(n - k)$, into which the binomial $(1 + x^n)$ is wholly divisible. The generator polynomial is selected to fit the length of the code sequence and error statistics, from among the polynomials given in tables. After the generator polynomial has been chosen and the respective code sequence has the form $a_0a_1a_2 \ldots a_{n-k}$, the cyclic code may be constructed in the form of a matrix having n columns and k rows. The first row of this matrix is formed from the sequence corresponding to the generator polynomial $q(x)$ padded with zeroes. For the second row, use is made of $q(x)x$ (shift one place to the right); for the third, $q(x)x^2$ (shift two places to the right), and so on:

$$\mathbf{A} = \left\| \begin{array}{llllll} a_0 & a_1 & a_2 \ldots & a_{n-k} & 000 \ldots\ldots & 0 \\ 0 & a_0 & a_1 \ldots & a_{n-k-1} \quad a_{n-k}00 \ldots\ldots & & 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots & & & & & \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots & & & & & \\ 0 \quad 0 \quad 0 & & 0 & 0a_0a_1 \ldots\ldots\ldots & & a_{n-k} \end{array} \right\| \qquad (3.19)$$

For example, in the case of a (7, 4) code ($n = 7$ and $k = 4$) and an appropriately selected generator polynomial, $q(x) = 1 + x + x^3$, the generator matrix will have the form

$$\mathbf{A} = \left\| \begin{array}{ccccccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array} \right\|$$

In cyclic codes, the information bits occupy k most significant positions in the code combination and the check symbols $n-k$ less significant positions. The generator matrix enables all code sequences to be formed by adding together the rows in all possible combinations (Table 3.3).

Table 3.3

Code sequences		Code sequences	
check bits	information bits	check bits	information bits
110	1000	011	1001
011	0100	110	0101
101	1100	000	1101
111	0010	010	0011
001	1010	100	1011
100	0110	001	0111
010	1110	111	1111
101	0001		

Since, however, each row of the generator matrix is a combination of the generator polynomial $g(x)$ and zeros, it is wholly divisible into this polynomial. This property enables the operation of coding and decoding to be reduced to the modulo 2 division of the information sequence by the generator polynomial; it is also shared by all other code sequences formed by combining the rows of the generator matrix.

A received code sequence can be checked for validity by noting if it is wholly divisible into the generator polynomial. If there is no remainder left, the code sequence is assumed to have been received correctly.

The check polynomial $h(x)$ can be derived by dividing the binomial $1+x^n$ into the generator polynomial $g(x)$. In our example,

$$
\begin{array}{l|l}
1+x^7 & 1+x+x^3 \\
\cline{2-2}
1+x+x^3 & 1+x+x^2+x^4 \\
\cline{1-1}
x+x^3+x^7 & \\
x+x^2+x^4 & \\
\cline{1-1}
x^2+x^3+x^4+x^7 & \\
x^2+x^3+x^5 & \\
\cline{1-1}
x^4+x^5+x^7 & \\
x^4+x^5+x^7 & \\
\cline{1-1}
\end{array}
$$

$h(x) = 11101.$

Cyclic codes may be instrumented, using feedback shift registers. A likely arrangement is shown in the block diagram of Fig. 3.13. As is seen, it uses a four-bit feedback register and an input switch, *Sw*. Let the code sequence applied to the coder be 0101. During the first four clock periods ($k = 4$), this sequence fills the register. Then the input switch changes to state *2* and

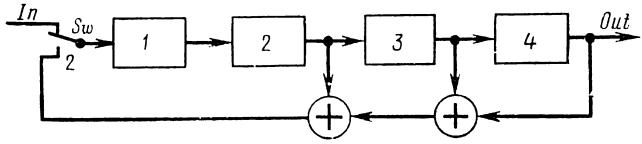

Fig. 3.13. Coding register

completes the feedback path. The next seven clock pulses ($n = 7$) cause the coder to form a seven-bit cyclic-code sequence (Table 3.4) and make it available at the output.

Table 3.4

Positions	*Output*
0 1 0 1	—
0 0 1 0	1
1 0 0 1	0
1 1 0 0	1
0 1 1 0	0
0 0 1 1	0
0 0 0 1	1
0 0 0 0	1

The output sequence is 1100101; as is seen, three check bits have been added to the information sequence.

A different coding scheme may be utilized. For example, the code sequence may be directly put into the communication channel and, in parallel, to a coder which forms additional check bits. Then the system will operate as follows (Fig. 3.14). To begin with, the signal coming from the control unit, *CU*, causes the commutator, *Com I*, to disconnect the output of the coding register, *CReg*, from the communication link. The information sequence is fed in parallel via commutator *Com II* into *CReg* and via *Com I* into the communication link. Then *Com I* connects the

output of *CReg* to the link, and the register operates $(n - k)$ more clock periods, delivering the check sequence into the link. Upon arrival of the next code sequence, the cycle of events is repeated all over again. At the receiving end, the transmitted sequence is applied to a decoder arranged to both detect and correct errors. Basically, this is a divider, but it is customarily

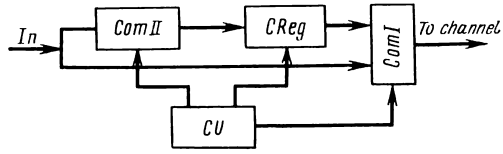

Fig. 3.14. Coder for a cyclic code, using division by the generator polynomial

called a decoding register. Functionally (Fig. 3.15), the complete system incorporates a decoding register, *DReg*, a buffer register, *BReg*, a parity-check-sequence analyzer, *PCA*, and an error corrector, *EC*. From the communication link, the code sequence goes

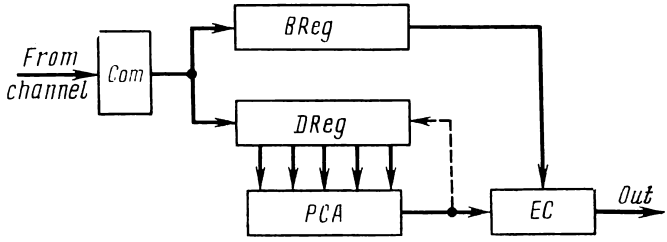

Fig. 3.15. Decoder for a cyclic code, using division by the generator polynomial

simultaneously to *DReg* and *BReg*. In n clock periods, *BReg* is filled full with the received code sequence, and *DReg* forms a parity-check sequence appropriate to the error vector. Thus, the buffer register acts as a kind of storage for the received sequence for the duration of division.

If the division of the code sequence stored in *BReg* by the generator polynomial $g(x)$ leaves no remainder (which signifies that no error has occurred during transmission), the sequence stored in *BReg* is made available to the recipient (while the check bits are erased). If, on the other hand, the division leaves a remainder, this is an indication of an error in transmission, and the parity-check sequence goes to the analyzer, *PCA*, whose function is to ascertain from the remainder what kind of error has occurred.

Then the code-sequence bits to be corrected are sequentially put out of the decoder via the error corrector. Just as an erroneous bit comes out of *BReg*, the parity-check analyzer feeds a correcting signal to the error corrector, and the latter removes the error by inverting the disturbed bit. In the case of single errors, this completes the operation of decoding. In the case of multiple errors, the correcting signal is at the same time applied to *DReg* (which is shown by the dashed line in the diagram), and the remainder produces the correcting vector for the bits to be corrected yet. In $2n$ clock periods, *DReg* must have been cleared to zero. If this does not happen, the error-correcting capacity of the code has proved inadequate.

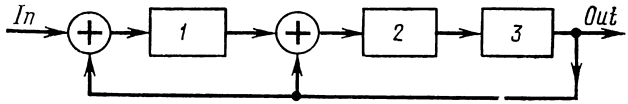

Fig. 3.16. Decoding register

It is seen from the diagram of the system that the input of the decoder should be disconnected for n clock periods in all cases examined above, so that the corrected sequence can be made available to the recipient. This is done by a switch, *Sw*.

In the case of purely error-detecting codes, the parity-check sequence analyzer is replaced by a simple error detector, and there is no error corrector (should an error be detected, the received sequence stored in the buffer register is deleted, and all elements of the decoder are reset).

The logic elements of the decoder operate in a sequence controlled by signals coming from the control unit (not shown in the diagram).

One of the likely arrangements for a decoding register appears in Fig. 3.16. The sole function of the register is to locate an erroneous bit. If there is no error, the parity-check sequence formed in the register after decoding will consist of only zeros. If the contents of the register is non-zero, this is an indication of an error.

The state transitions occurring in the decoding register when the code sequence 1100101 is received correctly are illustrated in Table 3.5.

In the case of an error in the code sequence (for example, 1101101), the decoding register will take up the state shown in Table 3.6.

Whenever a 1 appears in a particular position in the register, this is an indication that the received sequence is in error. In order to identify which place is actually in error, shift clock pulses are fed to the decoding register. The clock period during which a unity appears in the first position of the register and zeros in the remaining ones identifies the erroneous bit.

Table 3.5

Input	Positions	Output
1	1 0 0	0
0	0 1 0	0
1	1 0 1	0
0	1 0 0	1
0	0 1 0	0
1	1 0 1	0
1	0 0 0	
	No errors	

Table 3.6

Input	Positions	Output
1	1 0 0	0
0	0 1 0	0
1	1 0 1	0
1	0 0 0	1
0	0 0 0	0
1	1 0 0	0
1	1 1 0	0
	There is an error	

As an example, let the parity-check sequence 110 keep shifting in the register (Table 3.7).

Table 3.7

Clock period	1	1	0
1	0	1	1
2	1	1	1
3	1	0	1
4	1	0	0

The sequence 100 appears during the fourth clock period, which is an indication that the fourth bit in the received sequence is in error. It is corrected by inversion.

Majority decoding. Some cyclic codes can be handled by the *majority* decoding procedure which consists basically in the following.

As already noted, any code including a cyclic one may be specified in terms of a parity-check matrix which represents a system of check equations for the code in question. These check equations form what is called a *set of separated check equations* (or *rules*) defined as a set satisfying the following conditions: (1) the bit a_j should appear in each check equation; (2) the bit a_i $(i \neq j)$ should not appear in more than one check equation. When these conditions are satisfied, a single error will affect only one check equation which contains the mutilated bit, while two errors will affect not more than two check equations, etc.

Any set of equations can be solved first for, say, a_1, then for a_2, etc. Because the set is redundant, each of its independent variables can be evaluated in more than one way. The value of the bit a_j may be found by the majority approach, that is, the bit a_j is assumed to have the value satisfying the majority of check equations. The remaining bits are decoded by using the set of check equations for a cyclic code, because the check relations are satisfied by any codeword including that derived from the received sequence through a cyclic shift.

As an example, consider the cyclic (7, 3) code for which the generator polynomial has the form

$$g(x) = 1 + x + x^2 + x^4 = 11101$$

The parity-check matrix defining the set of separated check equations for a_1 has the form

$$\mathbf{H} = \begin{Vmatrix} 1101000 \\ 1000110 \\ 1010001 \end{Vmatrix}$$

The minimum Hamming distance in this case is $d = 4$.

The set of check equations may be written as

$$\left.\begin{array}{l} a_1 = a_2 \oplus a_4 \\ a_1 = a_5 \oplus a_6 \\ a_1 = a_3 \oplus a_7 \\ a_1 = a_1 \end{array}\right\} \quad \left.\begin{array}{l} a_2 = a_3 \oplus a_5 \\ a_2 = a_6 \oplus a_7 \\ a_2 = a_4 \oplus a_1 \\ a_2 = a_2 \end{array}\right\} \quad \left.\begin{array}{l} a_3 = a_4 \oplus a_6 \\ a_3 = a_7 \oplus a_1 \\ a_3 = a_5 \oplus a_2 \\ a_3 = a_3 \end{array}\right\} \tag{3.20}$$

With a single error, a wrong value for a_j is only given by the equation which contains the erroneous bit. As a consequence, the bit a_j may be determined by using a majority decision. If two

equations yield $a_j = 1$ and two more $a_j = 0$, this is a case of double-error detection.

A logic diagram of a decoder for the code in question appears in Fig. 3.17. It contains a decoding feedback shift register (flip-flops FF_1 through FF_7), a majority logic element M, three modulo 2 adders, mod-2, arranged to implement the set of check equations, (Eqs. 3.20), and an input commutator (or distributor) made up of two NAND gates tied together by an *OR* gate. Consider operation of the decoder.

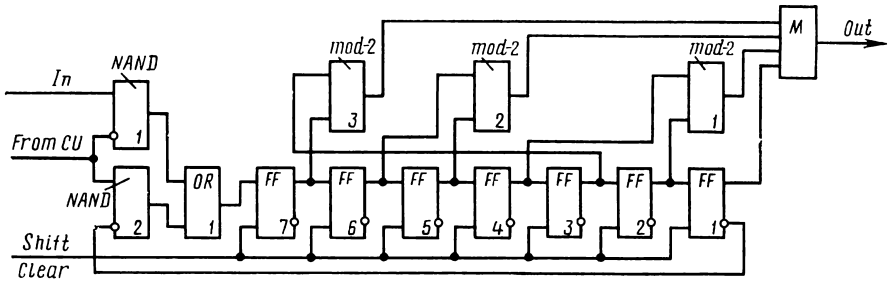

Fig. 3.17. Majority decoder

A signal coming from the control unit, *CU*, causes *NAND1* to open, and the information sequence fills the decoding register. After that, *NAND2* opens and *NAND1* closes. During the following clock periods, the modulo 2 adders evaluate a_j for each check equation and feed the values thus found to the majority logic element M which assumes the state representing the "majority" value of a_j and allows it to appear at the output. The decoding operation is complete after n cyclic shifts.

Let the code combination applied to the channel be 1100101. If the bit a_3 has been corrupted during transmission, the combination filling the register will be 1110101. During the next shift of the sequence through the register, feedback will be applied and the modulo 2 adders start operating. The sequence appearing at the input to the majority logic M will be 1101 and that at its output will be $a_1 = 1$. During the next shift the sequence at the input to M will be 0111 and that at its output, $a_2 = 1$. Then the input of M will accept 0001, and its output will present the corrected value of $a_3 = 0$, and so on.

Thus, after $n = 7$ clock periods, the correct sequence 1100101 will appear at the output. When this happens, all elements of the

decoder are reset in readiness to carry out the decoding cycle for the next code combination.

As regards cyclic codes adaptable to majority decoding, wider potentialities are offered by decoders with quasi-separated check equations.

A *set of quasi-separated check equations* is a set satisfying the following conditions: each check equation should contain several bits in the same linear form: the bit a_i should appear in not more than one check equation.

In the case of binary codes, a set of quasi-separated check equations is a separated one for the same subset of bits $a_{j1}, a_{j2}, \ldots, a_{jx}$, and so the parity-check matrix may have several columns containing only ones.

Consider a set of quasi-separated check equations as applied to the majority decoding of a (7, 4) code for which the generator polynomial has the form

$$g(x) = 1 + x + x^3 = 1101$$

The parity-check matrix for the bits a_1 and a_2 is

$$\mathbf{H} = \left\| \begin{matrix} 1110010 \\ 1100101 \end{matrix} \right\|$$

The rows of the matrix **H** form a set of quasi-separated check equations by which the modulo 2 sum of two bits, $a_1 \oplus a_2$, can be found in two independent ways. Extending this set to include a trivial relation, $a_1 \oplus a_2 = a_1 \oplus a_2$, and using a majority decision element, we obtain a decoder which yields the correct modulo 2 sum of the two bits $a_1 + a_2$ in the case of single errors:

$$\left.\begin{aligned} a_1 \oplus a_2 &= a_3 \oplus a_6 \\ a_1 \oplus a_2 &= a_5 \oplus a_7 \\ a_1 \oplus a_2 &= a_1 \oplus a_2 \end{aligned}\right\} \qquad (3.21)$$

A cyclic shift of Eqs. (3.21) produces sets of quasi-separated check equations for the sums of bits $a_2 \oplus a_3$, $a_3 \oplus a_4$, etc. It can be shown that this is sufficient for the bits of the transmitted message to be determined correctly.

If $\alpha_1 = a_1 \oplus a_2$ and the received sequence is shifted cyclically, the next clock period will cause the majority element to put out

$\alpha_2 = a_2 \oplus a_3$. Then for the bit a_2 we may write the following separated check equations:

$$\left.\begin{aligned} a_2 &= \alpha_1 \oplus a_1 \\ a_2 &= \alpha_2 \oplus a_3 \\ a_2 &= a_2 \end{aligned}\right\} \qquad (3.22)$$

These check equations yield the bit a_2 and all the other bits of the code combination. The resultant decoding delay for one clock period can readily be taken into account in the course of coding.

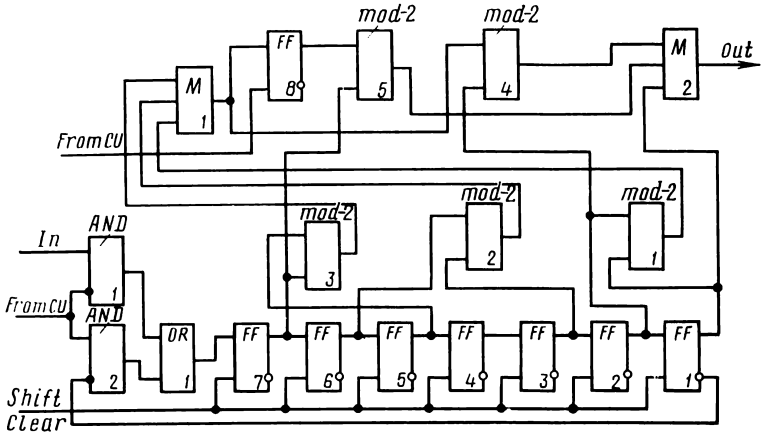

Fig. 3.18. Majority decoder using quasi-separated parity-check equations

A logic diagram of a decoder implementing the algorithm just examined is shown in Fig. 3.18. Consider the sequence of logic operations performed by this decoder.

When the *AND1* gate opens, the message sequence fills the decoding register using flip-flops FF_1 through FF_7. Then the *AND1* gate closes and the *AND2* gate opens; the message sequence is cyclically shifted and causes operation of modulo 2 adders *1*, *2* and *3*. The output of the majority element *M* presents the value α_1 which is applied to flip-flop FF_8. The bit a_1 is transferred from FF_1 to FF_7. During the next clock period, the output of M_1 will present the value α_2, and the output of M_2 the value of the bit a_2. The other bits will be evaluated in a similar manner during the succeeding clock periods. Thus, in $n + 1 = 8$ clock

periods, all bits of the received sequence are decoded and allowed to appear at the output. The mutilated bits are corrected in the same manner as in the example for separated check rules. A *Clear* signal clears the decoder in readiness for the next received code combination.

3.6. RECURRENT CODES

Codes in which code sequences are continuously coded and decoded are called *continuous* or *recurrent*.

In a recurrent code, check digits are placed among the message digits and formed in the course of coding by taking a modulo 2 sum of several message digits spaced a definite distance apart. In such codes, there are k information digits for n transmitted digits. The ratio of the number of information digits to that of continuously transmitted digits defines the redundancy of the code.

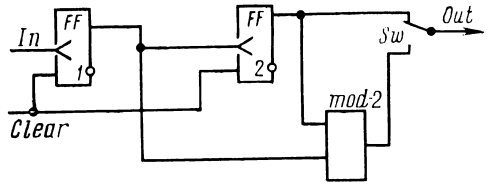

Fig. 3.19. Coder for a recurrent (1/2) code

A simple recurrent code is one with a redundancy $k/n = 1/2$. This is a code in which every check bit a_c is derived by taking a modulo 2 sum of two adjacent message bits a_m by the recurrent formula

$$a_{c2+2i} = a_{m1+2i} \oplus a_{m3+2i} \quad (i = 0,\ 1,\ 2\ \ldots)$$

A coder based on this equation is shown on the diagram of Fig. 3.19. It consists of a two-bit shift register, a modulo 2 adder, and a switch, *Sw*. It operates as follows.

After the first two bits of the message sequence has been placed in the register, the modulo 2 adder finds the first check bit a_{c2} in accordance with the recurrent equation. In the meantime, the switch *Sw* passes the first message bit a_{m1} and takes up the lower position, thereby allowing a_{c2} to reach the output. Then, the message sequence is shifted along the register, and the switch *Sw* takes up the upper position. Now comes the next message bit a_{m5}.

The register stores the second and third bits a_{m5} and a_{m3} of the message sequence. The modulo 2 adder finds the next check digit a_{c4}, and so on. In this way, the output delivers a sequence of check and message digits $a_{m1}a_{c2}a_{m3}a_{c4}a_{m5}$ . . . .

As is seen, each message digit contributes twice to the generation of check digits. For the first time, this is the leading check digit; for the second, this is the trailing check digit. In this configuration, the digits to which the same check rule applies stand next to each other, and only single errors can be detected. In order that bursts of errors can be corrected, guard spaces must be provided between adjacent blocks.

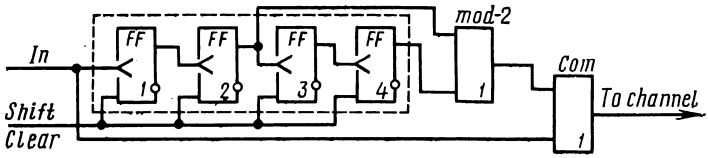

Fig. 3.20. Double-error correcting coder

Figure 3.20 shows the logic circuit of a coder capable of correcting double errors. A delay of two clock periods at the input to the register (equal in duration to the first two digits) extends the separation between check symbols to two digit positions, so that a capability is provided to correct not only double errors, but also bursts of up to four errors. For example, the message sequence $A_m = 101100111000$ fed to the coder produces a check sequence A_c which is formed by taking a modulo 2 sum of shifted message symbols:

$$\begin{array}{r|l} & 101100111000 \\ & \quad 101100111000 \\ \hline 00 & 100111110110 \end{array}$$

The synchronous commutator, *COM*, of the coder combines the message and check sequences

A_m	1	0	1	1	0	0	1	1	1	0	0	0	
A_c		0	0	1	0	0	1	1	1	1	1	0	1
A_Σ	1	0	0	0	1	1	1	0	0	0	0	1	1	1	1	1	1	1	0	1	0	0	0	1

and allows it to appear at the output. At the receiving end, the received code sequence is applied to a decoder such as shown in diagram form in Fig. 3.21.

The synchronous commutator, *COM*, of the decoder performs an operation which is the reverse of that carried out by its counterpart at the sending end, that is, splits up the received sequence into an message sequence A_m and a check sequence A_c. The message sequence then goes to a message register, *MReg*, and is uti-

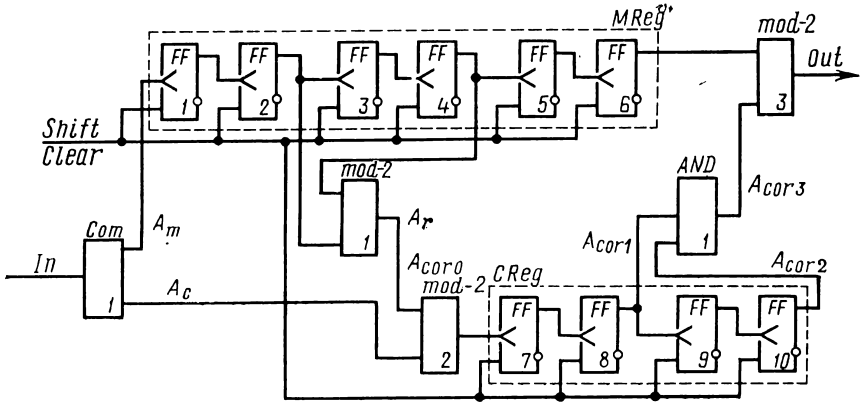

Fig. 3.21. Decoder for a recurrent code

lized to form a reference sequence, A_r, in modulo 2 adder *1* in much the same manner as this is done in the coder in order to determine the parity-check sequence A_c.

Then the reference sequence A_r and the check sequence A_c are serially fed to modulo 2 adder *2* which takes their sum (used as a correcting sequence, A_{cor0}). If the sum is zero (that is, if $A_r = A_c$), there is no error, and the message sequence A_m is made available to the recipient. If the sum is non-zero, the received sequence contains an error in either the message or the check segment. For example, if one of the check digits of A_c is unity instead of zero (underscored in the example), there will be a 1 in the same position in the correcting sequence:

A_r	0 0 1 0 0 1 1 1 1 1 0 1 1 0
A_c	0 0 1 <u>1</u> 0 1 1 1 1 1 0 1 1 0
A_{cor0}	0 0 0 1 0 0 0 0 0 0 0 0 0 0

If an error has occurred in a message position, two digits of A_r will be in error:

			1	0	1	1	$\underline{1}$	0	1	1	1	0	0	0
					1	0	1	1	$\underline{1}$	0	1	1	1	0	0	0
A_r	0	0	1	0	0	1	$\underline{0}$	1	$\underline{0}$	1	0	1	1	0
A_c	0	0	1	0	0	1	1	1	1	1	0	1	1	0
A_{cor0}	0	0	0	0	0	0	1	0	1	0	0	0	0	0

and, as is seen, unities appear in two positions of the correcting sequence A_{cor0}. To correct the errors, the mutilated bits of the message sequence A_m should be lined up with the unities in the correcting sequence A_{cor0}. For this purpose, the check register *CReg* forms shifted corrected sequences A_{cor1} and A_{cor2} which are then applied to the AND gate

A_{cor0}	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
A_{cor1}		0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
A_{cor2}			0	0	0	0	0	0	1	0	1	0	0	0	0	0		
A_{cor3}	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

At the output of the AND gate, a 1 will appear only in that digit position of the sequence A_{cor3} where unities concurrently appeared at the input.

The error is corrected in modulo 2 adder *3* by adding unity to the mutilated bit:

A_m	1	0	1	1	1	0	1	1	1	0	0	0
A_{cor3}	0	0	0	0	1	0	0	0	0	0	0	0
	1	0	1	1	0	0	1	1	1	0	0	0

Recurrent codes are simple, but they are excessively redundant which is a major limitation.

CHAPTER 4

DECODING OF REDUNDANT DIGITAL SIGNALS

4.1. DECODING BY SEQUENCE AND SYMBOL ESTIMATION

When redundant digital signals are received, a decoder converts them into code combinations of symbols representing elementary messages (characters of the message alphabet). This can be done in any one of two ways.

One is to estimate the entire signal received and to take a decision about which code combination to choose on the basis of such estimation. This approach may be called *decoding by sequence estimation.*

The other way involves two steps of decoding. To begin with, the received sequence is analyzed into elements in order to identify the symbols (ones or zeroes) used and, as a consequence, the code combination they form. Conversion of the received signal to a code combination reduces to demodulation and regeneration implemented by a circuit which may arbitrarily be called the *first decision circuit.* Then the code combination is converted to the most probable elementary message (character). This is done by a decoder which may arbitrarily be called the *second decision circuit.* This approach may be called *decoding by symbol estimation.*

Because the received signal is identified with a character of the message alphabet in two independent steps, decoding by symbol estimation does not utilize all information about the received signal. For the first decision circuit determines the posterior probability of each symbol on the basis of the received signal $z'(t)$ and selects one having the highest probability. The code combination thus formed, $y'_n y'_{n-1} \ldots y'_1$, is applied to the second decision circuit which identifies it with the most probable combinations $y_n y_{n-1} \ldots y_1$ on the basis of some predetermined criteria. Now, the parameters of the signal $z'(t)$ and the posterior probabilities of the symbols are neglected.

Decoding by sequence estimation utilizes all information about the received signal during signal reconstruction, but the decision circuit needed is fairly complex. This is why resort is sometimes made to a compromise which, although it does not utilize all of the information contained in the received signal, reduces its loss

in comparison with decoding by symbol estimation. As in the latter case, the circuits implementing this compromise approach reconstruct the received message in two steps, but the second step utilizes the information about the received signal $z'(t)$ as well. As an example, consider the circuit shown in Fig. 4.1. Extracted by the demodulator, *DEM*, the signal envelope is applied to the regenerator, *REG*, which decides whether the received symbol

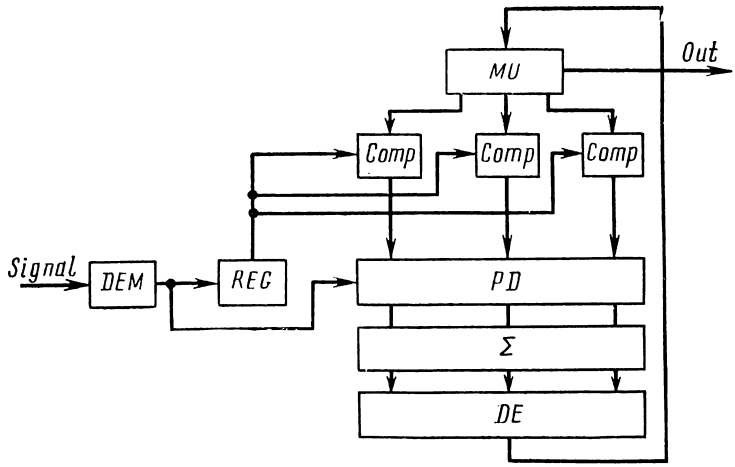

Fig. 4.1. Decoding of redundant digital signals

is a 1 or a 0 by counting the number of gating pulses. The maximum count of gating pulses for each symbol is α. The total number needed to make a decision as regards the symbol being reconstructed, including an uncertainty condition, is $2\alpha + 1$.

In the general case, whatever the state of the channel, the transmitted signal is subjected to noise and interference. With some approximation, it may be assumed that the signal distortions have the normal distribution with known parameters. We seek to determine the posterior probability P_j of correctly identifying the received signals given the probability P_i of error per symbol and the probability density of signal distortion in all likely states of the channel. To achieve this, we divide the entire decision area into two parts, Y_1 and Y_0. If the count of gating pulses falls in the area Y_1, a 1 is generated; otherwise, a 0 will be generated.

Let us designate the entire decision area as unity (Fig. 4.2), and partition it into two equal parts. Inside each area, we provide

$m+1$ validity gradations (each gradation contains C_j gating pulses), each uniquely identified with a 1 or a 0:

$$\frac{C_j}{2\alpha+1} \in Y_1 \qquad \text{if } 0 < \frac{C_j}{2\alpha+1} \leqslant 0.5$$

$$\frac{C'_j}{2\alpha+1} \in Y_0 \qquad \text{if } 0.5 < \frac{C'_j}{2\alpha+1} \leqslant 1$$

where $C'_j = \alpha + C_j$.

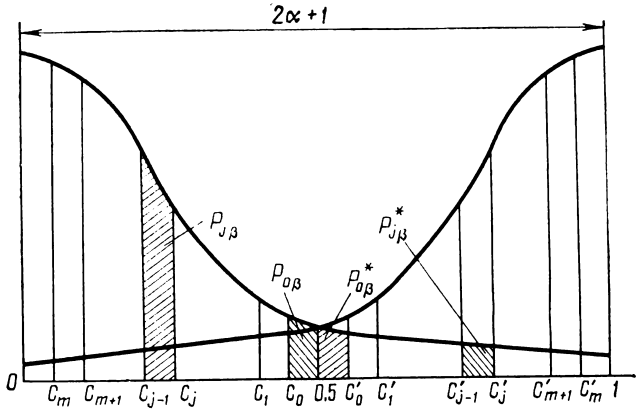

Fig. 4.2. Optimal decoding by sequence estimation

The probability that the symbol is received with a distortion corresponding to a given validity gradation is

$$P_{j\beta} = \Phi(h_j) - \Phi(h_{j-1}) \tag{4.1}$$

where

$$h_j = \frac{C_j}{(2\alpha+1)\,\sigma_i}$$

$$h_{j-1} = \frac{C_{j-1}}{(2\alpha+1)\,\sigma_i}$$

The probability that the received symbol is in error is

$$P^*_{j\beta} = \Phi(h'_j) - \Phi(h'_{j-1}) \tag{4.2}$$

where

$$h'_j = \frac{C'_j}{(2\alpha+1)\,\sigma_i}$$

$$h'_{j-1} = \frac{C_{j-1}}{(2\alpha+1)\,\sigma_i}$$

and $\Phi(h_i)$ is a normalized Laplace function of the form

$$\Phi(h) = \frac{1}{\sqrt{2\pi}} \int_0^h \exp(-h^2/2)\, dh$$

(usually given in tables).

Since a binary communication channel is symmetric, this reasoning applies equally to the other symbol.

From the values of $P_{j\beta}$ and $P^*_{j\beta}$ thus obtained, we may write the following likelihood relation:

$$(1 - P_j)/P_j = P_{j\beta}/P^*_{j\beta} \tag{4.3}$$

and the posterior probability of receiving the signal in error:

$$P_j = P^*_{j\beta}/(P_{j\beta} + P^*_{j\beta}) \tag{4.4}$$

Thus, at this stage the probabilistic detector, *PD* (see Fig. 4.1) ascertains the quality of reception of signals by determining the posterior probability of their reception in error.

Then the decoding operation proceeds as follows.

The regenerated symbols are serially fed to the comparators, *Comp*. At the same time, the comparators accept the respective digits of all legitimate code combinations from the memory unit, *MU*. The two sets are compared digit-by-digit, and the "coincidence" or ""non-coincidence" signal is fed to the probabilistic detector which corrects the assumed probability of error per symbol.

The posterior probability of error per symbol

$$P'_j = 1 - P_j \tag{4.5a}$$

when the two sets fail to match (a "non-coincidence" signal is generated), and

$$P'_j = P_j \tag{4.5b}$$

when the two sets match (a "coincidence" signal is generated).

The probabilities thus found are then applied to the adder, $\sum$, which determines the expectation M_n of error for each code combination compared. If we assume that comparison has revealed a match in l digit positions and no match in $n - l$ digit positions, then M_n for each step of comparison will be

$$M_n = P_N \frac{1}{n} \left[\sum_{\substack{j=m \\ l=0}}^{\substack{l=n \\ l=0}} P_{jl} + \sum_{\substack{j=0 \\ l=0}}^{\substack{l=n \\ j=m}} (1 + P_{jl}) \right] \tag{4.6}$$

where P_{jl} is the probability of error in the lth symbol received with the jth validity gradation, P_N is the probability that the kth code combination was transmitted, and m is the number of validity gradations.

The decision element, DE, estimates all M'_ns and selects the code combination for which $M_n = M_{n\,\min}$. This combination is then put out of the memory unit, MU.

In the case of optimum decoding, each received sequence is compared digit-by-digit with those stored in MU, and the combination for which the energy integral comes closest to that of the received sequence is assumed to be the correct one.

Practical application of optimal decoding involves a great number of complex conversions. Simplification is only possible with techniques where decoding utilizes more or less rough characteristics of a given realization of a random sequence (Wagner's detection scheme, symmetric erasure zone detection, etc.).

4.2. ERROR IMMUNITY OF DECODING BY SEQUENCE ESTIMATION

As regards error immunity, decoding by sequence estimation is often compared with decoding by symbol estimation on the basis of Fink's theorem. This theorem states that, assuming P_1 to be the probability of error per combination in decoding by symbol estimation, P_2 the probability of an uncorrectable error in decoding by symbol estimation, P_3 the probability of error per combination in decoding by sequence estimation, and P_4 the probability of an undetectable error in decoding by symbol estimation, then for any code we may write

$$P_1 \geqslant P_2 \geqslant P_3 \geqslant P_4 \tag{4.7}$$

The theorem may be interpreted as follows. In the case of redundant codes, decoding by sequence estimation has a higher error immunity than error-correcting decoding by symbol estimation, but a lower error immunity than error-detecting decoding by symbol estimation with an automatic request for repetition over the feedback channel. In the case of non-redundant codes, decoding by sequence estimation is equal to decoding by symbol estimation. To prove the theorem, consider the conditions under which code combinations will be recognized as erroneous in the various methods of decoding. In all cases, it will be assumed that the Hamming distance is d.

In decoding, by symbol estimation, a code combination will be recognized as erroneous if it differs in at least one digit position from that which matches it most closely, irrespective of whether the error can be corrected or not. Let this event be designated as A_1 and its probability as P_1. In error-correcting decoding by symbol estimation, an uncorrectable error occurs when $|d/2|$ positions fail to match. Let this event be designated as A_2 and its probability as P_2. Lastly, in error-detecting decoding by symbol estimation, an undetectable error occurs if noise or interference causes the received code combination to change to another legitimate combination. Let this event be designated A_4 and its probability, P_4.

It may be concluded from the foregoing that whenever the event A_4 takes place, the events A_2 and A_1 occur always. If the event A_2 takes place, the event A_1 occurs always. In other words, the set of events A_1 contains the subset of events A_2 which in turn contains the subset of events A_4. Therefore, their probabilities are in the following relation:

$$P_1 \geqslant P_2 \geqslant P_4 \tag{4.8}$$

These probabilities are equal only when $d = 1$, because the events A_1, A_2 and A_4 will then be coincident.

With decoding by sequence estimation, the received combination will be recognized as erroneous if the expectation of error within its length is $M_n = M_{n\,\min}$ and the combination contains any variants of errors. Let this event be designated A_3 and its probability, P_3. Obviously, all transitions to legitimate code combinations will fall in the category of undetectable errors. As a consequence, the set of events A_4 is a subset of the set A_3, and so

$$P_3 \geqslant P_4 \tag{4.9}$$

The two probabilities are equal at $d = 1$.

With decoding by symbol estimation, an uncorrectable error more than $(d/2)$-fold may be related equally to any legitimate code combination. With decoding by sequence estimation, however, the symbol probabilities yield additional information about the Hamming distance, which is equivalent to an increase in the error-correcting capability of the code and, conversely, to a reduction in the multiplicity of uncorrectable errors. Therefore, the events A_2 and A_3 may occur simultaneously and independently of each other. That is, they are partially intersecting subsets.

Also, the subset A_2 is always greater than the subset A_3. Their probabilities bear the same relation. Therefore, it is always that

$$P_2 \geqslant P_3 \tag{4.10}$$

The two probabilities are equal at $d = 1$.

The relations between the subsets A_1, A_2, A_3 and A_4 are depicted in Fig. 4.3.

By combining Eqs. (4.8), (4.9) and (4.10), we obtain the following inequality for each pair of code combinations the distance d apart:

$$P_1 \geqslant P_2 \geqslant P_3 \geqslant P_4 \tag{4.11}$$

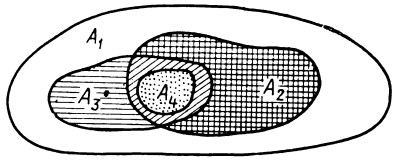

Fig. 4.3. Graphic representation of probabilities in the decoding of redundant digital signals

If the characteristics of the channel are known and the code to be used is selected, it is always possible to find the probabilities P_2 and P_4 and to estiblish the upper and lower bounds on P_3 that is, to determine the probability of error for decoding by sequence estimation.

4.3. WAGNER'S DETECTION SCHEME AND DECODING ON THE BASIS OF THE RELIABLE SYMBOLS

A distinction of Wagner's detection scheme is that a correcting code is only used to detect mutilated symbols and not their positions. Errors are corrected by comparing the posterior probabilities P_j of the received symbols, identifying the most unreliable ones, and correcting them. Wagner's detection scheme provides a rational approach to the utilization of a code's redundancy, because it offers a tool to correct all errors of multiplicity $t \leqslant (d/2)$. A block diagram of a decoder implementing Wagner's scheme is shown in Fig. 4.4.

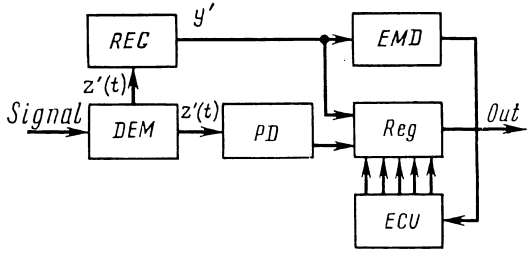

Fig. 4.4. Block diagram of a decoder

The decoder operates as follows. The incoming signal is demodulated by a demodulator, *DEM*, regenerated in a regenerator, *REG*, and detected in a probabilistic detector, *PD*, in much the

same way as in the circuit of Fig. 4.1 intended for decoding by sequence estimation. The received code combination along with its probabilistic characteristics is stored in a register, *Reg*. At the same time, the order of errors is established by parity checks. A signal from the error multiplicity detector, *EMD*, causes the corrector to generate an error-correcting vector for the register, *Reg*, where the most unreliable symbols are corrected. Then, the corrected combination is allowed to pass from the corrector to output.

Another decoding technique, also standing midway between decoding by sequence estimation and decoding by symbol estimation, is decoding on the basis of the most reliable symbols. In this case, decoding proceeds as it does in the arrangement shown in Fig. 4.1, except that the decision about the transmitted code combination is made on the basis of the most reliable symbols, n_1, rather than on all n symbols contained in the code combination. Since in any code combination, at least $d-1$ symbols may be erased without destroying the last chance to identify the various code sequences, the number of the most reliable symbols must lie in the interval

$$k \leqslant n_1 \leqslant n - d + 1$$

In view of the foregoing, the decoding procedure is initiated by selecting the number k of the most reliable symbols from the received code combination. From that point on, the decoding algorithm is similar to that used in decoding by sequence estimation, Eq. (4.6). The decoding procedure is complete when M'_k uniquely identifies one of the code combinations. Otherwise, one more out of the remaining most reliable symbols, $n-k$, is added to those previously selected, and all steps are repeated again.

In decoding on the basis of the most reliable symbols, it is possible to correct all $(d-1)$-fold errors, provided that $n-d+1$ correctly received symbols have a greater measure of likelihood than the incorrectly received symbols.

4.4. ERASURE DECODING

The difficulties associated with the implementation of decoding by sequence estimation have spurred the wide use of detection or decoding with two validity gradations or, which is the same, with a symmetric erasure zone. In symmetric channels, erasure decoding (Fig. 4.5) is accomplished by a two-threshold detector *DT* or

a detector wth two quantizing levels, and by the regeneration of the signal $z'(t)$ appearing at the output of a demodulator, *DEM*. The detection algorithm is as follows.

The signal $z'(t)$ is identified with a 1 if it exceeds some predetermined threshold C_1. If, on the other hand, it is below some threshold C_0 a 0 is generated. If $C_1 \geqslant z'(t) \geqslant C_0$, an uncertainty will exist as to what symbol was sent ($P_j = 0.5$), and an erasure signal, e, is generated. This is an indication that, if necessary, the information should be reconstructed by special rules.

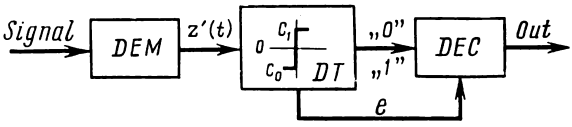

Fig. 4.5. Decoding using an erasure signal

The basic factors describing erasure decoding are the probability q_s that the signal is received correctly, the probability p_e of signal erasure, and the probability p_s that the received signal is in error. The relation between these probabilities is markedly dependent on the form of the first decision circuit and takes the form:

$$q_s = 1 - p_e - p_s \tag{4.12}$$

Decoding with erasure degenerates to decoding by symbol estimation when the erasure interval reduces to zero.

For non-redundant codes, erasure decoding of message sequences may be described as follows:

in the case of an independent error distribution, the probability of a code combination to be received correctly is

$$Q_s = (1 - p_s - p_e)^n \tag{4.13}$$

the probability of being erased for at least one symbol in the received combination is

$$p_e = 1 - (1 - p_e)^n \tag{4.14}$$

Consider the basic procedure for redundant codes in the case of decoding with erasure.

Information in the form of binary symbols and erasure signals appearing at the output of the first decision circuit is applied to a decoder, *DEC*. This case is analogous to decoding with two

validity gradations, if it is assumed that the erasure zone e lies anywhere between C_0 and C'_0.

From the view-point of simplicity, it is of interest to examine the correction of errors by erasure signals in conjunction with correcting codes. The length of erasure burst, $r = n - k$, determines the number of check digits used.

Parity-check rules for an n-unit code with k message digits are as follows:

$$a_{k+1} = a_1 \oplus a_{1+r} \oplus a_{1+2r} \oplus \ldots$$
$$a_{k+2} = a_2 \oplus a_{2+r} \oplus a_{2+2r} \oplus \ldots$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$a_n = a_r \oplus a_{2r} \oplus a_{3r} \oplus \ldots$$

For example, for a code having $k = 9$ and $n - k = 3$, $n = 12$ the following set of parity-check rules may be written:

$$a_{10} = a_1 \oplus a_4 \oplus a_7$$
$$a_{11} = a_2 \oplus a_5 \oplus a_8$$
$$a_{12} = a_3 \oplus a_6 \oplus a_9$$

If the probability of being received in error is expected to be highest for the erased symbol, it is possible to reconstruct any burst of erasures up to r long. For this purpose, each parity-check equation is analyzed by Wagner's method, and the erased symbol which requires correction will be in the equation that is not satisfied.

4.5. PROBABILISTIC DECODING OF GROUP CODES

Probabilistic decoding of group codes is simpler to instrument than decoding by sequence estimation. Those best suited to probabilistic decoding are low-density parity-check group codes.

A low-density parity-check (n, m, b) *group code* is an n-unit code specified by a parity-check matrix $\mathbf{H}_{n, m, b}$ having n columns with m ones in each column and b ones in each row (the remaining positions are filled with zeroes), such that $m < b$. In this matrix, $m - 1$ rows are linearly dependent, which implies that low-density parity-check codes are inseparable.

Parity-check equations (or rules) may be written in explicit form in terms of check digits. Provided that the above requirements are satisfied, all code combinations will be treated equally by the decoding procedure. That is, all symbols will be decoded

with the same degree of validity, on the average. This is because in an optimal low-density parity-check code each symbol appears in an equal number of parity-check equations and all parity-check equations contain the same number of symbols. For example, for a (6, 2, 3) code, the parity-check matrix is

$$\mathbf{H}_{6,2,3} = \begin{Vmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{Vmatrix}$$

In this matrix, only three rows are linearly independent. Therefore, this is a group (6, 3) code with $k = 3$ and $n - k = 3$. In order to select positions for check digits, let us write a parity-check matrix of the form

$$\mathbf{H}_{6,3} = \begin{Vmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{Vmatrix}$$

where all rows are linearly independent. If the first positions of all rows are occupied by 1's, the check digits should take up positions corresponding to the diagonal arrangement of 1's in the matrix columns. In our case, the check digits may take up the 3rd, 4th and 5th positions.

Therefore, the parity-check equations, in accordance with the matrix $\mathbf{H}_{6,3}$ will have the form

$$\left.\begin{aligned} a_3 &= a_1 \oplus a_2 \\ a_4 &= a_1 \oplus a_6 \\ a_5 &= a_1 \end{aligned}\right\} \tag{4.15}$$

Using the above equations, we may express each message symbol in explicit form:

$$\left.\begin{aligned} a_1 &= a_2 \oplus a_3 \\ a_1 &= a_4 \oplus a_6 \\ a_1 &= a_5 \end{aligned}\right\} \quad \left.\begin{aligned} a_2 &= a_1 \oplus a_3 \\ a_2 &= a_3 \oplus a_5 \\ a_2 &= a_3 \oplus a_4 \oplus a_6 \end{aligned}\right\} \quad \left.\begin{aligned} a_6 &= a_1 \oplus a_4 \\ a_6 &= a_4 \oplus a_5 \\ a_6 &= a_2 \oplus a_3 \oplus a_4 \end{aligned}\right\} \tag{4.16}$$

A decoding algorithm for low-density parity-check codes in the case of decoding by symbol estimation may be presented in the

form of a code tree. By this algorithm, all parity checks are carried out consecutively, starting with the first symbol. The symbols for which the number of unsatisfied equations exceeds some limit r are corrected. After that, the parity-check sequences are computed, using the new values of symbols. The procedure is repeated until all parity-check equations are satisfied. For the (6, 2, 3) code mentioned above, the decoding procedure may be depicted as a tree (Fig. 4.6) where the number of branches and

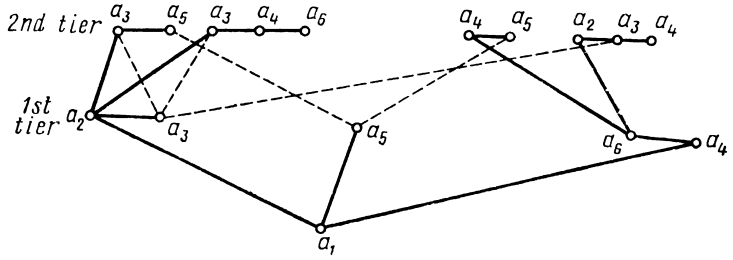

Fig. 4.6. Decoding tree for the (6, 2, 3) code

the manner in which they are connected represent the subset of parity-check equations starting with the symbol a_1, while the solid lines connect the symbols according to Eq. (4.16). The parity-check equations for the symbol a_1 use the symbols of the first tier. The parity-check equations for the symbols in the first tier use the symbol a_1 and the symbols in the second tier.

Let us set $r = 3$. After parity checks we find that none of the parity-check equations for the symbol a_2 are satisfied:

$$a_2 \neq a_1 \oplus a_3$$

$$a_2 \neq a_3 \oplus a_5$$

$$a_2 \neq a_3 \oplus a_4 \oplus a_6$$

and one equation each for a_1 ($a_1 \neq a_2 \oplus a_3$) and a_6 ($a_6 \neq a_2 \oplus \oplus a_3 \oplus a_4$). This is an indication that the symbol a_2 has been received in error and that it should be corrected. The other symbols are checked in a similar manner. This is decoding by symbol estimation, and it is applicable where parity-check equations are not numerous and the channel is symmetric.

Since decoding by symbol estimation is not an optimal scheme, it is more attractive to examine the probabilistic decoding procedure for low-density parity-check codes. For an insight into the procedure, we shall examine the decoding tree for a code using

three parity-check equations ($m = 3$) and four digits in each parity-check equation ($b = 4$), as depicted in Fig. 4.7. Here, the decoding procedure starts with some message symbol a_i. The lines radiating from it define a set of linear equations containing a_i, while the lines drawn from the first to the second tier represent the sets of parity-check equations containing the symbols of the first tier. The probabilistic detector estimates the posterior probabilities P_j for each of the received symbols. After the parity-check equations are solved for the symbol a_i, the new value of the

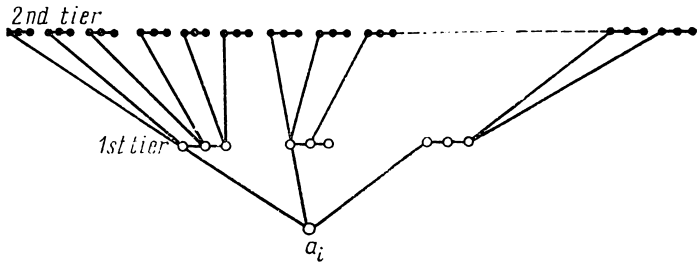

Fig. 4.7. Decoding tree for a code with $m = 3$ and $b = 4$

probability, P'_j is estimated. If most parity-check equations are not satisfied, $P'_j > P_j$, and $P'_j < P_j$ in the opposite case.

The decoding sequence is as follows. To begin with, P'_j is estimated for a_i. Then similar operations are performed for all symbols in the first tier on the condition that the probabilistic characteristics of the symbols in the second tier are known. When successful, this iterative procedure causes the probabilities for each symbol to tend either to zero or to unity. Therefore, on solving the set of equations for some symbol a_i, it will be corrected if the probability P'_j exceeds 0.5, and it will remain unchanged if the probability is less than 0.5.

This principle of decoding holds only when no loops form in the code tree. If, on the other hand, the same symbol appears more than once in the next tier of the code tree, loops form in it. Examples of loops are shown in Fig. 4.6 as dotted lines for the symbols a_3 and a_5. The symbol a_2 will be corrected unduly although the symbol a_3 has been received in error, and this will lead to the propagation of errors.

The decoding procedure may be simplified still more for cyclic codes. Consider a probabilistic decoding procedure for cyclic codes, using codes which allow majority decoding as an example.

Let there be a cyclic (7, 3) code with validity gradations γ_i. A logic diagram of an applicable decoder is shown in Fig. 4.8.

The binary symbols of the received code combination and the respective validity gradations are applied in parallel to the input of the decoder. The received code combination is loaded via *Input 1* into a code-combination register, *CReg*, and the validity gradations are written via *Input 2* into the validity gradation register, *VReg*. Then the first symbol is decoded. To begin with,

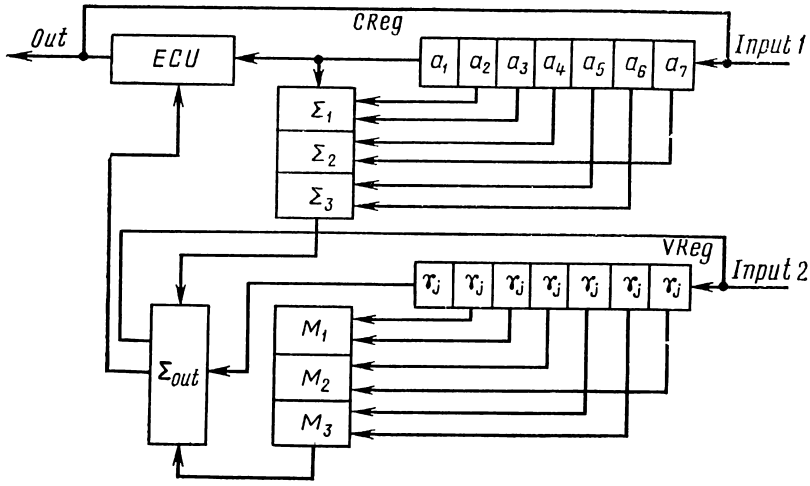

Fig. 4.8. Block diagram ot a probabilistic decoder for the (7, 3) cyclic code

parity checks are carried out in modulo 2 adders Σ_1 through Σ_3. The check results are applied to the output adder, Σ_{out}. At the same time, the probabilistic information about the symbols appearing in the parity-check equations is fed from *VReg* to majority logic elements M_1 through M_3. The majority logic elements identify the least validity gradation γ_i and apply it to Σ_{out} which, using some parity-check equations and check results, computes the new value of the validity gradation for the symbol being decoded. According to the result thus obtained, a correcting (inverting) signal is generated. From the error-correcting unit, *ECU*, the new value of the validity gradation is applied to *VReg* over the feedback channel so that it can be taken into account in decoding the next symbol.

The above steps are repeated sequentially for each symbol in the code combination being decoded. Before the next code com-

bination is applied, the decoder is reset. The probabilistic decoding of cyclic codes improves the validity of decoding message sequences because it utilizes the additional information about the received message in the form of validity gradations. In terms of error immunity, this decoding procedure of cyclic codes stands midway between Wagner's detection scheme and the probabilistic decoding of low-density parity-check codes. In comparison with the latter, however, it is simpler to instrument.

4.6. PROBABILISTIC-ALGEBRAIC ERROR DETECTION

As the order of errors in coded combinations increases, the use of algebraic error detection becomes increasingly more difficult. As a way out, attempts have been made to detect errors of low orders by algebraic techniques, and those of higher orders by probabilistic methods. Quite appropriately such combinations are called *probabilistic-algebraic*. Owing to them, it is possible to reduce the total redundance per code combination. One such method is based on estimating the error expectation and comparing it with some fixed value.

The error expectation over the length of the received n-unit code combination in systems using probabilistic detection and $(m+1)$ validity gradations may be found from the following relation:

$$M_n = (P_0 l_0 + P_1 l_1 + \ldots + P_j l_j + \ldots + P_m l_m)/n$$

where l_j is the number of symbols received with the γ_jth validity gradation.

By setting a specific threshold M for the current value of the expectation at which any received code combination will be recognized as an erroneous one, it is possible to detect errors varying in multiplicity without having to insert redundancy for their algebraic detection. The higher the threshold M, the greater the probability that an error of low multiplicity will be detected. As M increases, the probability of false alarm is increased.

An optimum value for the threshold M can be found by minimizing the probability of failing to detect errors, provided the probability of false alarm is maintained below some predetermined value. In the circumstances, the probability of error detection decreases with decreasing multiplicity of errors.

By inserting redundancy, it is possible to detect errors of low multiplicity by the algebraic method.

From the theorem for the multiplication of probabilities, the probability of failing to detect an error in a code sequence is

$$P_{Mk} = P_M P_k$$

where P_M is the probability of failing to detect errors on the basis of the threshold value, and P_k is the probability of failing to detect errors by the algebraic method.

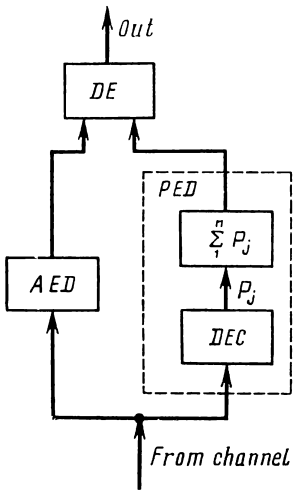

Fig. 4.9. Block diagram of an algebraic-probabilistic error detector

Calculations show that the probabilistic-algebraic method markedly reduces the probability that errors can pass undetected in comparison with the algebraic method, assuming the same amount of redundancy. This property becomes increasingly more noticeable as the length of code combinations increases.

The probabilistic-algebraic method is simple to instrument. A block diagram of an applicable arrangement is shown in Fig. 4.9. It includes an algebraic error detector, *AED*, implemented with feedback shift registers, and a probabilistic error detector, *PED*, incorporating a decoder, *DEC*, to estimate the probability P_j, and also an adder, $\sum_1^n P_j$, to determine the random value of the error expectation M_n for the code combination and to compare it with a preselected threshold, M. After each code combination is estimated, which leads to a delay relative to the instant when the last symbol is received, an output signal is generated, representing the decision whether there is or there is not an error in the received code combination.

CHAPTER 5

TWO-WAY DATA COMMUNICATION SYSTEMS

Apart from resort to error-correcting codes, data-communication reliability can be improved by extending the system to include a feedback or return channel. In block-diagram form, such a system containing a single message source, *MS*, and a single message recipient, *MR*, is shown in Fig. 5.1. The system incorporates a forward channel transmitter, *FWT*, a forward channel, *FWC*, a forward channel receiver, *FWR*, a feedback channel transmitter, *FBT*, a feedback channel, *FBC*, and a feedback channel receiver, *FBR*.

Feedback supplies information about the transmitted message, and the kind of noise and interference acting in the system, and enables appropriate measures to be taken.

In fact, feedback may be applied without a physical return channel, by using only the forward channel and time-division multiplex.

Among the merits of a two-way (or feedback) system is the fact that the desired signalling reliability can be achieved without resort to redundant codes. In non-feedback systems using error-correcting codes, it is usual to base the design on some averaged statistical data about the channel. With feedback, errors can be analyzed continually, so that the actual state of the channel can be ascertained during transmission and any amount of redundancy can be inserted at will so as to secure the desired performance. With a channel maintained in good repair, this approach reduces overall redundancy.

Feedback is especially effective where interference in both directions is uncorrelated or where interference in the feedback channel is markedly lower than that in the forward channel, so that the feedback channel has a greater degree of reliability. For example, in a space-to-ground (down) link the capabilities of the space-borne transmitter in terms of power output are severely limited. In contrast, the return channel from the ground-based transmitter to the spacecraft (the up-link) has greater capabilities and, as a consequence, a greater reliability.

Feedback systems may be divided into those with data feedback and those with decision feedback, according to where the transmitted data are tested for validity.

In *data-feedback systems*, this is done at the transmitting end.

In *decision-feedback systems*, this is done at the receiving end to a predetermined algorithm. According as the received message is correct or not, a decision is taken to dispatch the decoded message to the recipient (in which case an acknowledgement is sent over the return channel) or to request a repeated transmission. Quite appropriately, such systems are sometimes referred to as *automatic repetition request* (ARQ) *systems.*

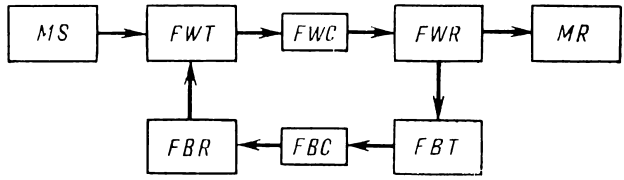

Fig. 5.1. Block diagram of a two-way (feedback) data communication system

Feedback may be applied before a decision is taken (that is, to that part of the system which lies ahead of the first decision circuit), after a decision is taken (that is, to that part of the system which extends as far as the second decision circuit), and both before and after the decision. In the first case, feedback is also applied to the communication link (the continuous or analog channel) and enables a decision to be made from an estimate of the continuously received (analog) signal. In the second case, feedback is applied to the greater part of the system and enables a decision to be made from an estimate of the discrete (or digital) sequence of code symbols after demodulation. The third case is self-explanatory.

A further subdivision of decision-feedback systems is into those with receiver interlock and those with address repetition.

In *receiver-interlock systems*, the detection of an error initiates a request for repetition, and the receiver is disabled, that is, rendered incapable of receiving further code combinations. Upon receipt of a request for repetition, the transmitter repeats the message starting at the data sequence received in error and up to the receipt of the RQ signal. In *address-RQ systems*, a request for repetition is only concerned with the addresses of the mutilated code combinations.

5.1. DATA-FEEDBACK SYSTEMS

In data-feedback systems, the received message is tested for validity by sending over the return channel to the sending end either the decoded sequence of symbols or a special signal derived from the received signal by an agreed rule.

Data-feedback systems show a very high degree of transmission fidelity, provided the level of noise and interference in the return channel is low. Among all other types of feedback systems they are simplest.

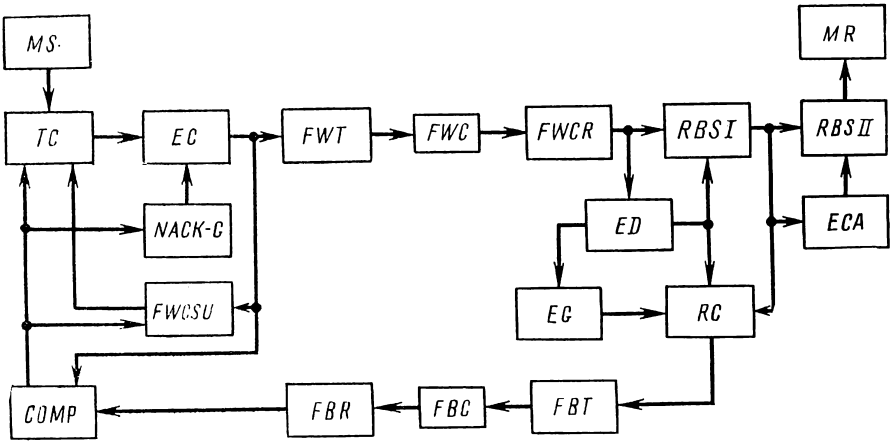

Fig. 5.2. Signal-flow diagram of a two-way (feedback) data communication system

The manner in which a data-feedback system operates will be clear from reference to the signal-flow diagram of Fig. 5.2. The message to be transmitted is applied from the message source, *MS*, via the transmitter commutator, *TC*, into an encoder, *EC*, which converts the simple code into a noise-immune one, whence the encoded message is sent by the forward-channel transmitter, *FWT*, into the forward channel, *FWC*. The code combinations appearing at the output of the encoder are at the same time fed to a buffer storage of the forward channel, *FWCSU*, and a comparator, *COMP*. The other input of the comparator accepts information about the transmitted code combination from the feedback-channel receiver, *FBR*. The comparator matches the two combinations. In the case of a perfect match, a "*Valid*" signal is generated; otherwise, an "*Error*" signal is produced. The "*Valid*"

signal causes the message source to send the next code combination into the forward channel. The "*Error*" signal stops the flow of messages from the message source, and a negative-acknowledge generator, *NACK*, generates a "Negative acknowledge" combination which is sent into the forward channel, while the forward-channel storage unit is connected to the input of the encoder in order to repeat the combination received in error.

The received code combinations are processed as follows. They go simultaneously to an error detector, *ED*, and the first buffer storage of the receiver, *RBS1*. In the absence of errors, the error detector generates a "*Valid*" signal, and the code combination goes simultaneously to the second buffer storage of the receiver, *RBS2*, an erase combination analyzer, *ECA*, and via the receiver commutator, *RC*, to the feedback-channel transmitter, *FBT*. If the received combination is not an erase combination, it is made available to the message recipient, *MR*, from the second buffer storage of the receiver. If this is an erase combination it causes the second buffer storage of the receiver to be cleared. In the case of an error, the error detector generates an "*Error*" signal which causes the erase generator, *EC*, to erase the contents of the first buffer storage of the receiver. At the same time, the negative-acknowledge generator forms a *NACK* combination to be sent over the return channel, *FBC*.

The same logic may be employed in the case of a system using a simple code. The amount of equipment needed is then drastically reduced, but transmission fidelity suffers markedly because there is no error detector and the probability of an erase combination being transformed into a legitimate one becomes equal to that for mutilated code combinations. It is therefore customary in such systems to use a simple error detector based on a single parity-check code.

Further insight into operation of a data-feedback system is provided by the timing diagram of Fig. 5.3. Here T_0 is the time required to process and generate signals, T_1 is the time taken by the signal to propagate, Π_i is the capacity of the buffer storage, A_{fw} and A_{fb} are the numbers of combinations of, respectively, the forward-channel transmitter and feedback-channel receiver at the sending terminal, B_{fw} and B_{fb} are the numbers of code combinations of, respectively, the forward-channel receiver and feedback-channel transmitter at the receiving terminal. The directions of transmission are indicated by arrows (solid arrows designate the forward direction, and the dotted lines, the return or feedback

direction). The numerals stand for the numbers of code combinations.

When the transmitted and received combinations are the same, the forward channel conveys the next code combination. Otherwise, it conveys a "*Negative acknowledge*" signal S_{na} following which the code combinations transmitted previously are repeated. At the receiving terminal, the *NACK* signal, S_{na}, causes erasure

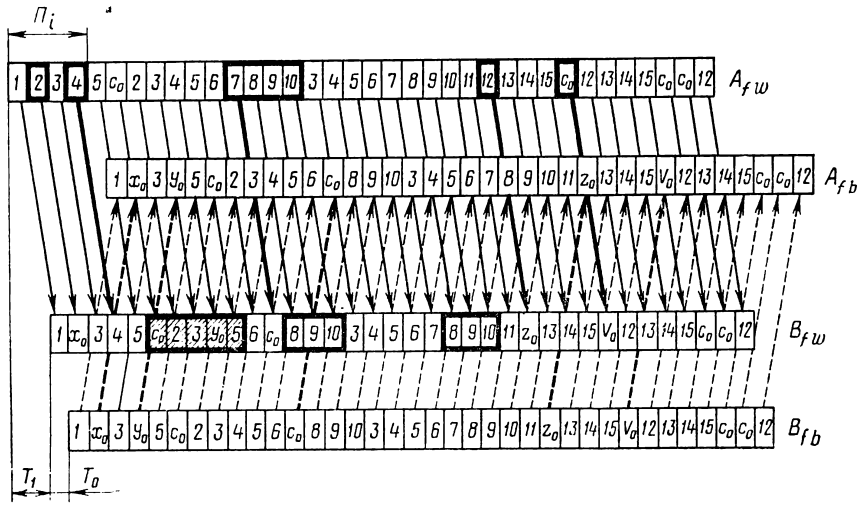

Fig. 5.3. Timing diagram of a two-way (feedback) data communication system

of all mutilated combinations that precede it and equal in number to the capacity of the buffer storage. For example, should combination *2* be erroneously transformed into some combination x_e, the sending terminal will detect the mutilation not until after a time interval equal to $2(T_0 + T_1)$, that is, after combination *5* has been transmitted. When the error is detected, a *NACK* signal is sent, and the group of $\Pi_i = 4$ preceding combinations are sent along. At the receiving terminal, the *NACK* signal and the four previously stored combinations are erased (shown by cross-hatching in the figure). During the repeated transmission, the previously transmitted combinations keep arriving at the transmitting terminal. Some of the latter may likewise be mutilated (for example, combination *4* has been corrupted into y_e), but no repeated *NACK* signal need be sent, because this combination is repeated all the same, and the decision about its validity is made on the basis of the last reception.

Should a *NACK* signal arrive over the return channel, the forward channel will repeat $2\Pi_i$ combinations (say, combinations *3* through *10*), but no *NACK* signal will be sent out. As a result, errors may occur. For example, should combination *7* be corrupted in the course of transmission into a *NACK* signal, the detection of this error will cause the forward channel to repeat transmission from combination *3*, code combinations *8* through *10* will be registered twice, and this will produce a burst of errors.

If a code combination V_e be received instead of a *NACK* signal, the forward channel will convey two *NACK* signals followed by Π_i combinations preceding the *NACK* signal. As a result, the receiving terminal will erase $2\Pi_i$ working combinations, the first *NACK* signal, the V_e combination, the second *NACK* signal, and mutilated combination *12* which precedes V_e and has caused the sending of the *NACK* signal transformed into V_e.

From reference to the timing diagram of a data-feedback system, the following basic limitations may be noted:

(1) mutilation of code combinations sent over the feedback channel makes it necessary to repeat both the mutilated and the correct preceding combinations;

(2) transformation of legitimate combinations into a *NACK* signal produces a large bursts of errors whose correction complicates the system appreciably;

(3) mutilation of the *NACK* signal causes the erasure of large blocks of data in the buffer storage of the receiver. Because of this, buffer storage has to have an increased capacity and made more elaborate, or the *NACK* signal must be protected against mutilation.

The drawbacks listed above have limited the use of data-feedback systems.

For the most part, uncorrectable errors in a data-feedback system occur because the transmitted code combinations are transformed into one another and also because these combinations are corrupted into a *NACK* combination or vice versa. The probabilities of such mutilations may be reduced to a negligibly low value by the use of an error-detecting code. However, this will require a larger buffer storage and reduce the data rate.

The capacity of buffer storage for data-feedback systems is mainly governed by the signal propagation time and may be found by the equation

$$\Pi_{data} \geqslant (2T_1/n\tau_0) + 3 \qquad (5.1)$$

where $2T_1$ is the time in which a signal travels in the forward and feedback channels, n is the number of symbols per combination, and τ_0 is the duration of an elementary pulse (unit element or unit interval).

As is seen, the time required to process the combination is neglected.

A basic parameter of a data-feedback system is the signalling rate, V_{data}. For a single parity-check code and the known symbol error probability p,

$$V_{data} = V_0 \frac{n-1}{2n} (1-p)^{2n(\Pi_i+1)} \tag{5.2}$$

where V_0 is the signalling rate for a simple (non-redundant) code.

The factor 2 in the denominator takes care of the fact that the feedback channel is not utilized for the transmission of useful information. Account is also taken of the fact that a code combination is presented to the recipient only after Π_i combinations following it have been received.

In data-feedback systems, errors in the messages made available to the recipient occur mostly owing to the mirror conversion of code symbols, that is, owing to the fact that a mutilated symbol is corrected through the action of mutilations in the return channel.

5.2. DECISION-FEEDBACK SYSTEMS USING NON-REDUNDANT CODES

Decision-feedback systems may utilize probabilistic information about code symbols in several ways. Accordingly, they may be classed into those where a request for repetition (RQ) is generated for each symbol and those where this is done for a combination as a whole. In the former case, wide use is made of erasure detection or two validity gradations; in the latter, an arbitrary number of validity gradations may be used (usually, four or five), while the state of the channel is monitored on the basis of the threshold specified for a random quantity — the error expectation, that is, by detecting errors within the length of the registered code combination probabilistically.

Symbol-RQ systems. A distinction of such systems is that, upon receipt of a *NACK* signal, each symbol will be repeated until a positive acknowledgement comes.

Operation of a symbol-RQ system will be clear from the signal-flow diagram of Fig. 5.4.

The message source, *MS*, feeds a message in binary symbols through the transmitter buffer storage, *TBS*, to the forward-channel transmitter, *FWT*, which converts and sends the binary characters into the forward channel, *FWC*. At the receiving terminal, the inbound signals are demodulated by a demodulator, *DEM*, and analyzed by a double-threshold detector, *DTD*, which generates binary symbols and an erase signal. If a symbol is not erased, it goes to the receiver buffer storage, *RBS*. If the symbol

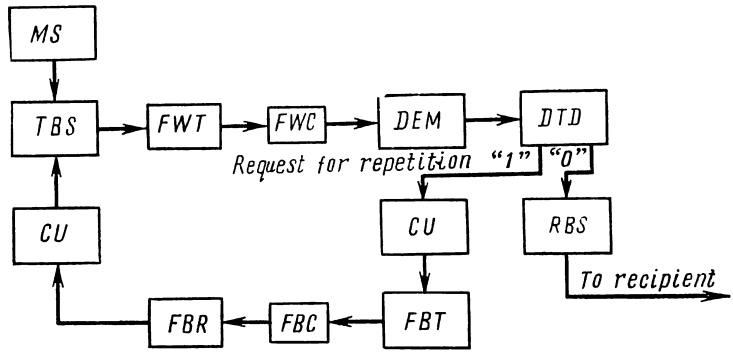

Fig. 5.4. Signal-flow diagram of a symbol-RQ data communication system

is erased, the control unit, *CU*, causes the feedback-channel transmitter, *FBT*, to send over the feedback channel, *FBC*, an RQ signal which initiates the repeated transmission of the erased symbol.

In such a system, the average probability p_r of receiving a mutilated code combination may be estimated as

$$p_r = np_s/(1 - p_e) \tag{5.3}$$

where n is the number of symbols per combination, p_s is the symbol error probability, p_e is the symbol erasure probability, $1/(1 - p_e)$ is the average number of RQ signals per symbol, and $p_s/(1 - p_e)$ is the average probability that a symbol is received in error;

$$q + p_e + p_s = 1 \tag{5.4}$$

where q is the probability that a symbol has been received correctly;

$$V_r = V_0(1 - p_e) \tag{5.5}$$

where V_0 is the signalling rate for a simple code.

The above relations apply to an ideal feedback channel and a negligible time needed to transmit and receive signals. If these conditions are not satisfied, the performance of a symbol-RQ system is greatly impaired.

Combination-RQ systems. In such systems, the first decision circuit incorporates a probabilistic detector with an arbitrary number of validity gradations. The inbound message signals are processed in two ways. Firstly, they are regenerated and converted to short pulses; secondly, the validity gradations γ are computed and added together by an adder and compared with a

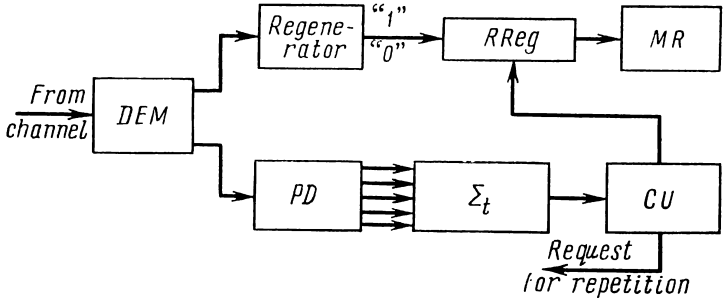

Fig. 5.5. Block diagram of a combination-RQ data communication system

fixed threshold. If the sum of the validity gradations γ over the entire length of the code combination exceeds the threshold, a request is generated for the repetition of the code combination involved.

In block-diagram form, the receiving terminal of a combination RQ system using a non-redundant code is shown in Fig. 5.5. The message coming from the communication channel is passed through a demodulator, *DEM*, regenerated to a "0" and a "1", and fills the receiving register, *RReg*. At the same time, the probabilistic detector, *PD*, estimates each message symbol. This estimation yields validity gradations which are added together in a threshold adder, $\sum_t$ and compared with some threshold value, *M*.

If the threshold value is not exceeded after estimation of the entire combination, the control unit, *CU*, releases the receiving register, and the received code combination is made available to the message recipient, *MR*. If the sum of all validity gradations is equal to or exceeds the threshold *M*, the control unit requests

the repetition of the code combination involved, the message already stored in the receiver register is erased, and the latter is emptied in readiness to receive the combination to be repeated.

The transmitting terminal of a combination-RQ system does not basically differ in operation from those of the systems already examined.

Combination-RQ systems have a higher transmission reliability than symbol-RQ systems because both each message element and the code combination as a whole are estimated more accurately. Also, there is a marked increase in sending rates, especially over variable-performance channels.

A very important variable in the design of such systems is the optimal value of the threshold M. The threshold adder $\sum_t$ acts as a probabilistic error detector. To minimize the maximum likely risk (defined here as the total damage due to the probability of failure to detect errors and false alarm probability), it is advantageous to use the minimax criterion.

The signalling rate in combination-RQ systems using an ideal feedback channel is

$$V_c = V_0(1 - p_M) \tag{5.6}$$

where p_M is the probability of erasing a code combination on the basis of the threshold M.

A very important property of such systems is that the fidelity of data transmission over the return channel can be improved materially.

5.3. RECEIVER-INTERLOCK SYSTEMS

As an example, consider the receiver-interlock system recommended by the CCITT for H.F. radio telegraphy and telemetry.

As a rule, in designing a receiver-interlock system, it is sought to utilize both the forward and return channel for signalling. This also applies to the transmission of repeat decisions. Therefore, to obtain an insight into the logic underlying the operation of such a system, it will suffice to examine operation of only one terminal subset at any end of a system. A block diagram of a terminal using symbol-by-symbol check and an error-detecting and error-correcting code is shown in Fig. 5.6.

From a sending teleprinter, TP_1, code combinations are applied to the input storage unit, ISU_1, whence they go to a buffer storage, BS, and a commutator, COM.

The signals reaching the receiver, *RCVR*, are decoded by a decoder, *DEC*, and channelled via ISU_2 to TP_2 and also to a request analyzer, *RQA*, and an error analyzer, *EA*.

In the absence of RQ signals and errors in the incoming combinations, the commutator *COM* connects ISU_1 to an encoder, *ENC* thereby giving clearance to send out the code combination.

If there is an RQ signal or a mutilated code combination, the commutator *COM* connects *BS* to *ENC* so that the former can

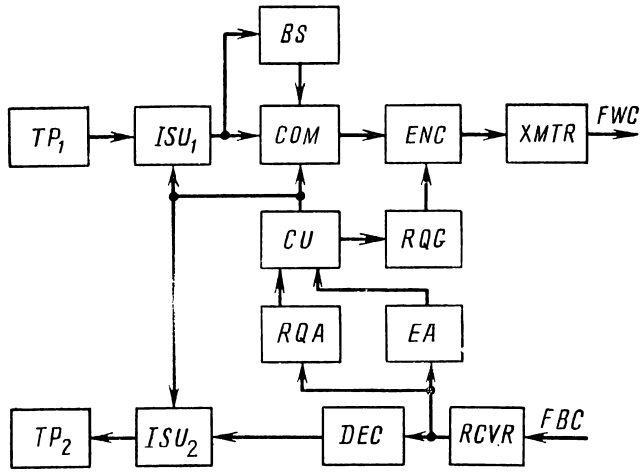

Fig. 5.6. Block diagram of an interlocked data communication system

repeat the previous code combination it stores. At the same time, the control unit, *CU*, disconnects the output of ISU_2, lest it should register erroneous information, and causes the request generator, *RQG*, to generate an RQ signal.

In the case of mutilation, errors may occur in one of the two channels (the forwards channel, *FWC*, or the feedback channel, *FBC*), in both channels, or in the RQ signal. As a result, a single RQ or a repeated RQ signal may be generated, as is depicted in the timing diagram of Fig. 5.7.

When errors occur in only one channel, the system operates as follows. Suppose that the combination a_3 transmitted from terminal *A* by transmitter A_0 has been corrupted into a forbidden combination x which is received by receiver B_0 of terminal *B*. The decoder will disclose the error. An appropriate signal from the decision unit will cause the buffer storage, *BS*, of the receiver

B_0 to delete three combinations (the system in question uses a single RQ), while transmitter B_1 will send out to terminal A a "*Negative acknowledge*" signal, S_{na}, followed by all combinations stored in *BS*. At the same time, an interlock will disable receiver B_0 (its input will be blocked) for an appropriate time interval (in our example, for the duration of three clock periods). The received *NACK* signal will disable receiver A_1 at terminal A for the

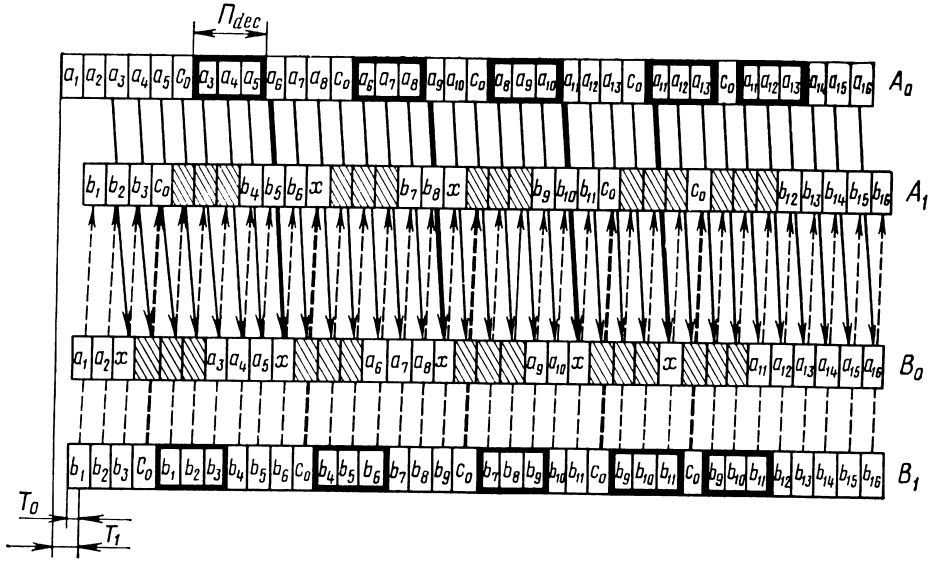

Fig. 5.7. Timing diagram of an interlocked data communication system

same time interval, and transmitter A_0 will send out the *NACK* signal, followed by the combinations a_3, a_4 and a_5 requested to be repeated.

Should code combinations be corrupted in both channels at the same time, receivers B_0 and A_1 will be disabled at both terminals upon discovery of errors for the duration of three clock periods, and transmitters A_0 and B_1 will send out *NACK* signals and the three requested combinations stored in *BS*, each over its own channel. The repeated combinations are erased from *BS* by the *NACK* signal.

Should the *NACK* signal sent by transmitter B_1 at terminal B be corrupted into a forbidden combination x which is received by receiver A_1, the system will repeat the *NACK* and the three previously transmitted combinations.

The worst case is failure to detect an error in the S_{na} combination. In such a case, terminal A will continue sending without repetition, and this will entail the loss of the three combinations deleted from the buffer storage, BS, of the terminal's receiver.

The logic of system operation in the case of multiple mutilation in the repeated combination is similar to that already discussed.

Thus, as long as a decision-feedback system using a duplex channel is operating normally, RQ signals are sent, erroneous combinations are deleted and repeated, and the terminal receivers are disabled both ways.

From the foregoing it follows that in a receiver-interlock system the signalling rate is mainly decided by the time for which the receivers are disabled and also by the probability p_{det} of a detectable error in the code combination, determined in turn by the quality of the communication channel.

The disable time depends on the capacity of buffer storage. If the propagation time exceeds the duration of a code combination and the RQ signal has the same duration as the code combination, three more combinations might be received during the interval between the detection of an error and arrival of the repeated combination. Therefore, if we ignore the time spent to process the signal, the capacity of buffer storage in terms of the number of stored code combinations will be

$$\Pi_{dec} \geqslant 2\,[T_1/(n\tau_0) + 1] \tag{5.7}$$

where Π_{dec} is the capacity of the transmitter buffer storage in a decision-feedback system.

The probability p_d that the receiver will not be disabled depends on whether there are errors in the last Π_{dec} combinations:

$$p_d = (1 - p_e)^{\Pi_{dec}} \tag{5.8}$$

where p_e is the probability of error in a code combination.

Therefore, assuming that a duplex receiver-interlock system has channels of the same quality, the signalling rate for an (n, k) code will be

$$V_{dec} = V_0 k\,(1 - p_e)^{2\Pi_i + 1}/n \tag{5.9}$$

In contrast to data-feedback systems, data are transmitted both ways.

The probability of presenting an erroneous code combination to the recipient is equal to the probability that the (n, k) code

fails to detect t-fold errors and may approximately be expressed as

$$p_k = \frac{1}{2^{n-k}} \sum_{t=\lambda+1}^{t=n} C_n^t p^t q^{n-t} \tag{5.10}$$

where λ is the order (multiplicity) of errors whose detection is guaranteed.

Sometimes, receiver-interlock systems utilize an error-correcting and error-detecting code. In such systems, an RQ signal is sent only when uncorrectable errors appear. In all other respects, the systems follow the same logic as that described above. With feedback arranged as outlined, the signalling rate is somewhat higher, but the count of equipment increases markedly.

A major limitation of receiver-interlock systems is the need to repeat large blocks of data even when there are no errors. This demerit is especially felt where data are to be transmitted at high rates and over long distances. The capacity of buffer storage has then to be markedly expanded, and this drastically slows down the signalling rate. This shortcoming can be avoided, if feedback is based on the address-RQ principle.

5.4. ADDRESS-RQ SYSTEMS

A distinction of address-RQ systems is that data are transmitted in large blocks, each containing a great number of code combinations. The system receiver incorporates a memory unit with a capacity sufficient to store all of each block and having a separate register for each code combination. The combinations received without errors fill appropriate registers, while erroneous combinations are erased, and the respective registers remain vacant. After all code combinations of a given block of data have been received, the addresses of all erroneous combinations are sent back over the return channel. The transmitting terminal registers these addresses, and the transmitter repeats the respective combinations upon an appropriate signal from the control unit. This sequence continues until no error is any longer detected in the transmitted block of data, and these are presented to the recipient.

As an example of an address-RQ system, consider the transmission of a block of data containing ten code combinations (Fig. 5.8).

When the control unit, *CU*, of the transmitting terminal generates a "*Read*" signal, the message source, *MS*, feeds a block of data containing ten code combinations in a parallel code into the transmitter buffer storage, *TBS*. At the same time, the control unit causes a "*Start-of-message*" signal to be sent to the receiving terminal. After that, the transmitter buffer storage feeds data as *k*-digit combinations to the encoder, *ENC*. The latter encodes each combination in an error-detecting code and sends them into the forward channel, *FWC*. When they reach the receiver, *RCVR*, the

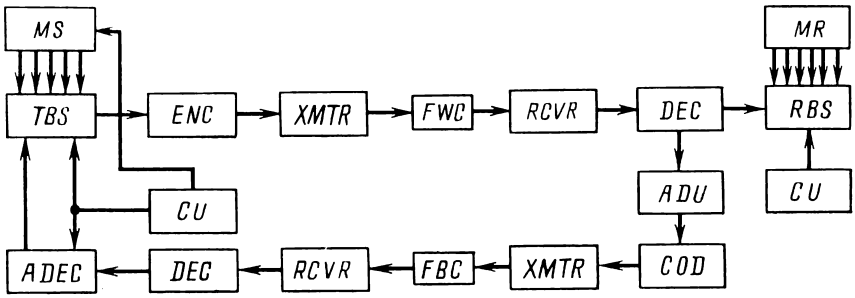

Fig. 5.8. Block diagram of an address-RQ data communication system

incoming code combinations are tested by the decoder, *DEC*, for errors. The message digits received without errors are written in the registers of the receiver buffer storage, *RBS*, at their respective positions (addresses). The addresses of mutilated combinations are noted by the address decision unit, *ADU*, and the combinations proper are erased. After all of the block of data has been received, an RQ combination is generated, converted into an error-correcting code and relayed over the feedback channel, *FBC*, to the transmitting terminal. For example, if in a message of ten combinations, the first, third and sixth have been corrupted, the address combination will have the form 1010010000 (the ones represent the positions of the mutilated combinations). In the absence of mutilation, the combination will be made up of only zeros. Upon reception, the address combination is decoded by the decoder, *DEC*, and applied to the address decoder, *ADEC*. The latter generates an appropriate signal which causes the transmitter buffer storage, *TBS*, to read out the applicable combinations. After appropriate conversion, these combinations are relayed to the receiving terminal. The above chain of events is repeated until no errors occur any longer. After the entire block of data

has been delivered to the recipient, a zero address combination is sent to the transmitting terminal, thereby causing the message source, *MS*, to put out the next block of data.

A major advantage of address-RQ systems is a reduction in the time wasted in repetitions and a practically complete independence of line length, provided a large-capacity buffer storage is used. These advantages grow in proportion as the signal propagation time increases.

However, address-RQ systems suffer from material disadvantages. Above all, they require a larger count of equipment, they cannot practically operate in real time, and specific errors will arise, should an address combination be received in error. This is why the use of address-RQ systems may be warranted only on long lines having a low probability of faithful data reception.

5.5. ADAPTIVE FEEDBACK SYSTEMS

A distinction of adaptive feedback systems is that they utilize practically all information existing at the channel output so as to adapt the decision-making procedure to changing line characteristics. This is why such systems widely use statistical decoding techniques. The choice of a particular statistical decoding technique in conjunction with a feedback channel depends on the quality of the communication channel used, the requirements for the fidelity of transmitted data, and the count of equipment permitted. If the receiver has a memory to store the statistical characteristics of the symbols appearing at the channel output, less time will be needed to find an optimal decoding procedure for each code combination received in error, provided there is a feedback channel. In the circumstances, the feedback channel may be used to adapt the decoding procedure purposefully and, as a consequence, to control redundancy and signalling rate in a flexible manner, as changes in the signalling conditions may require.

In systems using statistical detection and storage of statistical information about symbols at the channel output, the transmitted combination having the form $x_1x_2 \ldots x_i \ldots x_n$ will be registered by the receiver as a combination of the form $y_1y_2 \ldots y_i \ldots y_n$ with the probability P_j of error in identifying each symbol, where $j = 0, 1, \ldots, m$ (m is the number of fixed values of P_j). In such a case, it is possible to select from the received code combination a number m of sub-blocks in each of which the symbols will have

the same error probability P_j (Fig. 5.9), that is

$$\left.\begin{array}{l} y_{01}, y_{02}, \ldots, y_{0r_0} \in P_0 \\ y_{11}, y_{12}, \ldots, y_{1r_1} \in P_1 \\ \cdots\cdots\cdots\cdots \\ y_{m1}, y_{m2}, \ldots, y_{mr_m} \in P_m \end{array}\right\} \qquad (5.11)$$

where $r = 1, 2, \ldots, r_j$.

Obviously, whenever errors are detected, it is necessary to write r_j parity-check equations for the symbols received with a maximum symbol error probability, P_j. For this purpose, a signal

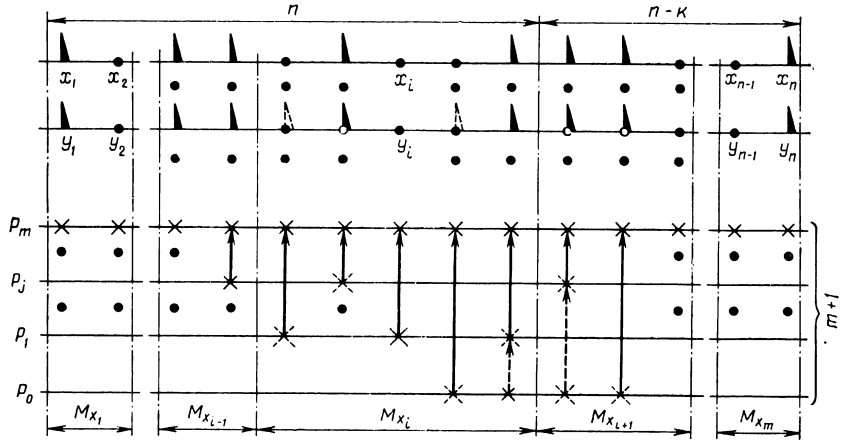

Fig. 5.9. Timing diagram of an adaptive feedback data communication system

generated in accordance with the parity-check rules should be sent to the transmitting terminal over the feedback channel. After the receiver has registered r_j parity-check symbols, the decoder should re-compute the probabilistic characteristics for each of the l symbols appearing in the r_j parity-check equations, and correct, at the same time, the respective characters in the code combination on the basis of the results thus obtained.

This type of decoding procedure may be implemented in a way similar to that used for the statistical decoding of cyclic codes. The logic of the decoder should preferably be implemented according to the following simplified algorithm:

$$\left.\begin{array}{l} P'_{il} \sim l\,(P_{j,\,l-1})_{\min} \\ l_{OM} \sim l\,(P_{jl})_{\max} \end{array}\right\} \qquad (5.12)$$

that is, the new value of the symbol error probability should be taken equal to the nearest lower one from among those appearing in the parity-check equation, and the error should be ascribed to the symbol received with the maximum value of P_j, irrespective of whether the parity-check equation has been satisfied or not.

Thus, in the case of errors, r_j symbols registered with the minimum posterior probability need be checked. The simplest decoding procedure will be obtained if each parity-check equation contains no more than one of the symbols to be checked. Since in real communication channels such symbols come in bursts, the above requirement can be satisfied through the use of a code in which each symbol to be checked is a modulo 2 sum of message digits shifted through r_j symbols relative to one another and to a character being checked. Then the parity-check matrix will have the form

$$\mathbf{H}_{r_j;\, n} = \left\| \begin{matrix} h(x_j) \\ xh(x_j) \\ x^{r_j-1}h(x_j) \end{matrix} \right\| \tag{5.13}$$

It has no sense to transmit such a matrix over the feedback channel.

In order to cut down the time during which the feedback channel is occupied, the actual redundancy and error probability, it is important that such a matrix have a minimum number of symbols and a simple structure, but at the same time represent the parity-check rules used.

The problem is to match a parity-check polynomial $h(x_j)$ to the expected error vector $\mathbf{E}_j$ so that each of the parity-check equations will contain no more than one non-zero element out of r_j and also, so that $h(x_j)$ will have a minimum degree. This requirement can be satisfied if we consecutively multiply $\mathbf{E}_j$ by the parity-check matrices $\mathbf{H}_{r_i,\, n}$ for which the degree of the parity-check polynomial may range from $r_i = r_j$ to $r_i \leqslant n/2$. Therefore, in accordance with (5.13), we get

$$\mathbf{H}_{r_j,\, n} = \mathbf{E}_j \otimes \mathbf{H}_{r_i,\, n} \tag{5.14}$$

where $r_i = r_j;\ r_{j+1},\ \ldots,\ r_a \leqslant n/2$, and r_a is the degree of the parity-check polynomial such that each parity-check equation will contain no more than one non-zero element out of $\mathbf{E}_j$.

Multiplying the expected error vector $\mathbf{E}_j$ by each row of the parity-check matrix $\mathbf{H}_{r_a,\, n}$ and writing the result in a row, we

obtain the parity-check matrix $\mathbf{H}_{r_a, n}$ in a compact form

$$\mathbf{C}(r_a) = [a_0, a_1, \ldots, a_{r_{a-1}}]$$

It is an easy matter to show that $\mathbf{C}(r_a)$ fully satisfies the basic requirements for parity-check equations to match the expected error vector $\mathbf{E}_j$: the parity-check matrix in compact form is equal in length to the degree of the parity-check polynomial $h(x_a)$, just necessary to check all r_j's, the number of units is equal to that of the symbols to be checked, and the digit positions taken up by the ones identify the parity-check equation involved.

The transmitter encoder should operate in accordance with the adopted parity-check algorithm. The check symbols may be obtained by multiplying serially or in parallel the code combination being decoded by the parity-check matrix formed in accordance with the algorithm coming over the feedback channel. When $a_{r_a} = 0$, the code combination need not be multiplied by the parity-check matrix.

In address-RQ systems, the entire block of n symbols in the transmitted code combination is decomposed into $(m+1)$ sub-blocks, each containing $d = n/(m+1)$ symbols. The probabilistic characteristics make it possible to compute the expectation of error for each sub-block:

$$\left.\begin{array}{l} y_{01}, y_{02}, \ldots, y_{0d} \in M_{x_0} \\ y_{11}, y_{12}, \ldots, y_{1d} \in M_{x_1} \\ \cdots\cdots\cdots\cdots \\ y_{m1}, y_{m2}, \ldots, y_{md} \in M_{x_m} \end{array}\right\} \qquad (5.15)$$

The expectation of error for each sub-block is

$$M_{x_i} = \sum_{l=1}^{l=n/(m+1)} (1 - Q_j) = \sum_{l=1}^{l=n/(m+1)} P_j \qquad (5.16)$$

If errors are detected, it will be more likely that they are concentrated in the sub-blocks received with the maximum error expectation, $M_{x_j \max}$. Therefore, the RQ-signal should be coded to give the numbers of the corrupted sub-blocks. Thus, a system using address-RQ and statistical decoding may be treated as one with an algorithmic feedback such that $l > 1$, while the principles and techniques of decoding are similar to those examined earlier.

Here, the probability that the recipient will be presented an erroneous code combination is estimated by (5.3), while the signalling rate, by analogy with decision-feedback systems, will approximately be

$$V_{M_k} = V_0 \frac{k_M}{n} (1 - p_e)^{2(\Pi_{dec}+1)} \tag{5.17}$$

where k_M/n is the fractional reduction in signalling rate due to statistical-algebraic error detection.

It is to be noted that the amount of additional information required to decode a combination received in error will, in each particular case, be a random quantity dependent on the order of errors in the combination being decoded and the probability that the received symbols are identified correctly. Computation shows that this amount will be smaller than in the previous case. This is why adaptive-feedback systems show faster signalling rates, especially over channels with high error rates. However, such systems require a greater count of equipment.

5.6. DATA-COMMUNICATION EQUIPMENT

In the early days of data communication, use was mainly made of rented (or leased) switched public telegraph networks (known under the name "Telex" in Europe and "TWX" in the United States). These networks still remain in service because they are relatively inexpensive to build and run, especially where data traffic is limited, although their reliability and data rate are not adequate.

Advances in data input-output and processing facilities have led to far more stringent requirements as regards transmission reliability and data rates. As a result, wider use is now being made of telephone (voice-grade) and wideband channels.

In order to implement the data transmission techniques discussed above with sufficient fidelity and to link users to distant computers, special-purpose facilities have been developed, referred here to as *data-communication equipment*. According to the physical channel used, it may be low-rate, using telegraph channels; medium-rate, using voice-grade (telephone) lines; and high-rate, using wideband channels.

By international agreement, various data rates have been adopted for these channels. More specifically, telegraph channels can operate at 50, 100 and 200 Bauds; telephone lines at 200, 600

and 1200 Bauds, and also at 1800, 2400, 3000, and 3600 Bauds, and higher speeds obeying the law 600×2^n, where $n = 3, 4, 5, \ldots$

Present-day data-communication equipment incorporates signal-conversion facilities,error-control devices, and ancillaries, such as automatic ringing and answerback devices.

A *signal converter*, is that part of a data communication terminal which converts digital data into analog signals and back, such as a *modem* (an integral combination of a modulator and a demodulator). A *modem* is a self-contained part of a data-communication equipment, both physically and functionally (Fig. 5.10).

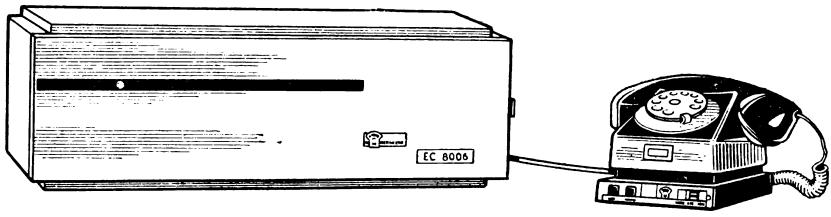

Fig. 5.10. Type EC-8006 modem

The choice of the modulation and receiver to be used in a modem is decided mainly by the characteristics of the associated communication channel. The modem and the channel together determine the data rate attainable. The characteristics of some Soviet-made modems are given in Table 5.1.

Table 5.1

Type designation	Data communication equipment	Channel	Data rate, bits/s	Modulation	Transmission
Modem-200	8001, 8002	Rented, switched	200	FM	Sequential
Modem-1200	8005, 8006	Same	600, 1200	FM	Same
Modem-2400	8010, 8011	Rented	2400	PM	Same

An *error-control unit* serves to minimize the effect of errors occurring during data transmission. Physically, this unit is built along the same lines as a computer. For ease of installation at the user's premises, the receiving and transmitting sections are usually built into a common console (Fig. 5.11). The electronic

Table 5.

Type designation	Data communication equipment	Data rate bits/s	Components
УЗО-50	Akkord-50	50, 100	Transmitter-receiver
УЗО-1200	Akkord-1200	600, 1200	Transmitter-receiver
УЗО-1200	8121	600, 1200	Transmitter, receiver; transceiver
УЗО-2400	8131	1200, 2400	Transmitter, receiver; transceiver

part widely uses standard circuit components, semiconductors, integrated circuits and printed wiring. The characteristics of some Soviet-made error-control units are listed in Table 5.2.

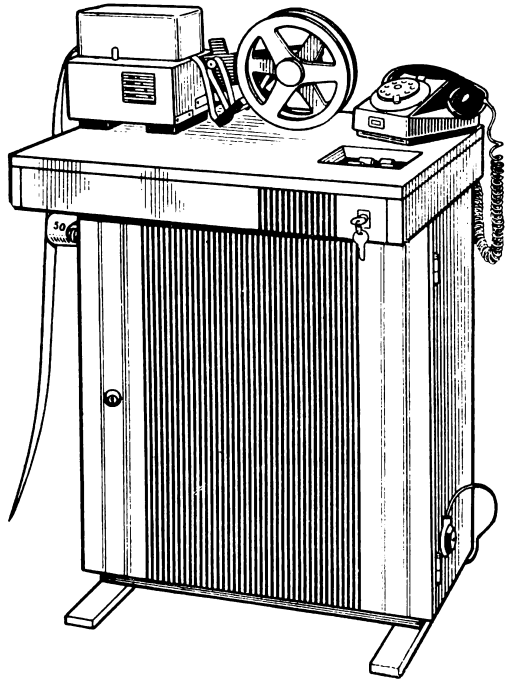

Fig. 5.11. Type 8122 error-control unit and terminal

Complete with data input/output facilities, *data-transmission equipment* makes up a data-transmission terminal. This may be a sending terminal, a receiving terminal or both.

Consider the Soviet-made **Akkord** series of data-communication equipment.

Akkord-50. This is an example of low-rate data-communication equipment for half-duplex (to and fro) working over switched or rented telegraph channels. Data can be put into line at 50 bits/s when using the No. 2 International Telegraph Code or 100 bits/s in the 7-bit weighted code to USSR Standard GOST 13052-67. In the latter case, appropriate functional bits are added to each code combination, and the latter is divided into two five-bit combinations.

As a way of improving transmission fidelity, the equipment uses decision feedback and a cyclic code with a generator polynomial of the form $1 + x + x^3 + x^6 + x^{12} + x^{13}$.

For the exchange of data between two points, each should have an Akkord-50 terminal of its own. Data to be exchanged are first keyed on punched tape, the latter is read by a tape reader in order to convert the data into appropriate electric signals in a parallel code, and the coded signals are finally put into the data-communication equipment. As an alternative, data may directly be keyed in, using a roll printer. At the receiving terminal, the data are presented in printed form by a similar roll printer or keyed on tape by a type ПЛ-150 tape punch. As a further alternative, the data-communication equipment may be interfaced to other input/output facilities or directly to a computer.

The data-communication equipment is connected in a switched public telegraph network and utilizes its facilities. In operation through an automatic telegraph exchange, a connection between two parties is set up with the aid of an apparatus which does the ringing, clearing and dialling. The maximum length of such lines is limited by the resistance of the cable conductors used, supply voltages and currents available and required for operation.

A typical terminal includes a distributor, a memory unit, an encoder, a decision unit, a power unit, and a control console.

The distributor is one of the major functional elements of the equipment. Basically, it operates in two modes, "*On-line transmit*" and "*On-line receive*".

In the "*On-line transmit*" mode the distributor generates signals that form start and stop characters; these characters are inserted between every five units of each message sequence sent into line, a space for "*Start*" and a mark for "*Stop*".

In the "*On-line receive*" mode, the distributor maintains synchronism with the distributor at the opposite end of the line and

generates signals to select message bits. In both modes, the distributor causes the data to shift through the registers and generates signals that control the operating cycle of the equipment.

The memory unit is built around an eight-bit shift register.

The encoder forms a cyclic code, provides connection with the channel, and generates shifting and timing signals. The heart of the encoder is a code translator which divides each sequence of message elements by the generator polynomial.

The decision unit houses the principal logic elements of the equipment.

The power unit converts a supply voltage of 220 V, 50 Hz, single-phase in all the necessary rectified and stabilized voltages.

The control console carries six keys with which the operator can run the equipment manually. These keys are labelled "*Data transmit*", "*Data receive*", "*TLG exch*", "*RPT*", "*End of transmission*", and "*Fault*".

As is seen, the data-communication equipment can be operated in any one of three modes: "*TLG exch*" (telegraph exchange), "*Data transmit*" (transmission of data), and "*Data receive*" (reception of data).

In the "*TLG exch*" condition, data are exchanged in accordance with the operational routine of the switched public telegraph network. In fact, the error-control unit takes no part in the procedure.

The "*Data transmit*" and "*Data receive*" conditions can be selected with the appropriate keys and a toggle switch or by a signal from an external circuit. Data to be transmitted are fed to the code translator which uses a noise-immune code, to the communication channel, and are also stored in the memory unit. The data coming from the line first enter the decoder which tests them for likely errors, and are read into the memory unit. If there are no errors, the memory unit delivers the message to the recipient, and sends an "*Acknowledge*" signal over the return channel to notify the transmitter that the message has been received correctly. In the case of an error, the decoder generates a "*Negative acknowledge*" signal which prevents the memory unit from reading out the incorrect data. On receiving a "*Negative acknowledge*" signal, the transmitter repeats the previously transmitted block of data. The probability of error in this system is one character in three million.

Akkord-1200. This is a medium-rate data-communication equipment. It is designed for high-fidelity handling of digital data over

switched and leased telephone (voice-grade) circuits within a city telephone system. Data are transmitted by a synchronous serial method at 600 or 1200 bits/s (75 or 150 eight-unit characters per second).

Data can be read in and out in serial blocks of 240 (or 112) binary units each, using a five-, six-, seven-, or eight-bit parallel code.

The equipment uses decision feedback and a cyclic code with a generator polynomial of the form $1 + x^5 + x^{12} + x^{16}$.

The fidelity of transmission is one corrupted character in a million.

The punches and readers are the same as in the Akkord-50 equipment. The data-communication channel is set up with a Modem-1200 and an Akkord-1200.

The error-control unit incorporates a transmitter, a memory receiver, a control unit, an interface, and a power unit.

The transmitter houses the following components:

a pulse generator which supplies clock pulses, control signals and addresses at which a particular character should be written or read in the memory unit;

a control signal unit, to receive, convert and generate exchange and control signals for the modem and to output;

a data-block number generator, to assign one of three serial numbers to the next data block;

a memory control unit, to generate data write and read signals;

a return-channel signal analyzer, to identify whether the signal is an RQ (request for repetition) or an ACK (acknowledge);

a range control, to set and correct the teleprinter range and to actuate alarms in the case of faults in the line;

a data input control unit, to receive the data to be relayed, to convert the parallel into a serial code, to write data in memory, to check the received character for parity, and to generate data-block identity signals;

an encoder, to form check bits in the cyclic code. The check bits are formed by multiplying a polynomial which is a sequence of an identity symbol and data block by x^{16}, and dividing the product into the generator polynomial;

a supervisory alarm, to indicate any faulty conditions in the functional elements of the transmitter.

The receiver consists of:

a synchronizer, to generate clock and control pulses;

a range control, to generate the RANGING signal in the case of initial and intermediate adjustment of the printing range;

an RQ or ACK signal generator for the return channel;

a decoder, to detect errors in the received data block. Errors are detected by noting if the division of the received sequence by the generator polynomial leaves a remainder. If the remainder is zero, the data block has been received correctly;

an input register, to convert the serial data coming from the modem into a parallel presentation;

a functional symbol detector, to decode the expected and previous data-block identifiers;

a data-block cyclic number generator, to enable the data blocks to be received in the correct sequence, should a fault occur in the return channel;

a circuit generating signals, to permit transfer of data from the modem to receiver, and from the receiver to output;

supervisory circuits operable in case of troubles in the receiver.

The memory unit receives, stores and delivers data. Its capacity is sufficient to store 64 nine-bit code combinations. Characters are written and read out in a parallel code. The memory unit can operate in one of three modes: write, read, read and restore. Physically, the memory unit consists of a core stack, a diode address decoder, and a write-read signal generator.

The control and monitor unit comprises a control keyboard and a light display panel. It provides a means for turning on power, selecting particular modes of operation, and actuating alarms in the case of malfunctions.

The power unit converts 220-V, 50-Hz supply into stabilized rectified voltages appropriate to particular circuits.

The equipment has a choice of several operating modes, including the initial margin (printing range) adjustment, data transmit and receive, data-block repeat, intermediate margin adjustment, end of transmission, and monitoring.

During the initial margin adjustment, the transmitter and receiver are set ready to process data continuously. An appropriate signal from the margin control circuit causes the input register of the receiver data input control unit to store 0101, which is the sync block identifier. This identifier is then routed via the encoder to the modem. From the receiver modem, data are routed to the receiver decoder operating as a 16-bit shift register. The register outputs are coupled to a decoder whose signal causes the control pulse generator to reset.

The data transmit/receive condition is the main operating mode of the equipment. It commences upon arrival of the "*Data set*" signal which causes the input control unit to generate an "*Input sync*" signal and data entry to be performed. If data are entered serially, the "*Input sync*" signal is maintained for a time interval sufficient to enter 240 (or 112) message symbols, after which it is terminated. The next signal may come only after a time interval necessary for the check bits of the current data block and the identifier of the next to be entered in the modem. If no "*Data set*" signal comes by the end of this interval, no "*Input sync*" signal will be generated, and the message segment of the data block will be padded with a dummy eight-bit combination, 01101001.

In the case of parallel entry, data are caused to be written into the input register of the data input control unit by a "*Data set*" signal. The "*Input sync*" signal indicates that the current character string has been received and the next one may appear on the input wires. Each code combination entered in parallel presentation is subjected to an even parity check.

The incoming data are grouped into blocks each containing 260 (or 132) bits. Of them, four are used as the block identifier, 240 (or 112) are message bits (arranged into 30 or 14 eight-bit combinations, respectively), and sixteen are check bits formed according to the generator polynomial. At the sending terminal, these blocks are fed into the modem as a sequence of binary code pulses.

At the receiving terminal, the incoming blocks are decoded and entered into the memory unit. Any corrupted data blocks are deleted and a request is sent for their repetition over the return channel in the form of an RQ signal. If all blocks have been received correctly, an "*Acknowledge*" signal is sent back over the return channel, while the received data are made available at the output.

Without waiting for an "*Acknowledge*" signal, the transmitter sends out the next block and retains the previous one in its memory unit. Should an RQ signal come, the transmitter will repeat the blocks stored in the memory unit. The memory unit has a capacity sufficient to store two contiguous data blocks. The two data blocks stored in the memory and the third being transmitted have the following identifiers: A (0011), B (1001), and C (1100).

The data-communication terminal reads out data serially as blocks of 240 (or 112) binary symbols. This is done only if an "*Output sync*" signal has come from the computer interface,

which signifies that the computer is ready to accept the message part of the data block. As a data block read out, a "*Data set*" signal is generated.

In the case of parallel output, if a data block has been received correctly, the five- (six-, seven- or eight-) unit code combination of the first character is set in the receiver output register, and the "*Data set*" signal is sent to the computer, thereby enabling the character to be read out. If the computer has received a given character correctly and is ready to receive the next, an "*Output sync*" signal should be fed to the terminal. Arrival of this signal causes the code combination of the next character to be set in the output register, and so on.

If no "*Output sync*" signal comes from the computer, data output is interrupted (the "halt" condition) until the signal comes. In the meantime, the character involved is stored in the output register, an RQ signal is sent back to the sending terminal, and no further data block can be received from the transmitter.

Parallel output is accompanied by an even parity check.

In the "repeat" mode, all data relayed to the modem, including the block identifier, are written into the memory unit in a parallel eight-bit code and stored there until an "*Acknowledge*" signal comes over the return channel.

If no "*Acknowledge*" signal comes, the input control unit generates no "*Input sync*" signal, the next data block is not sent, and two data blocks are transferred from the memory unit to the modem for repetition.

The transmitter resorts to the intermediate ranging mode if an RQ signal is generated for every out of nine transmitted data blocks in succession. In such a case, the data block following the sync block is read out of the memory unit along with the previous cyclic identifier and repeated together with the sync block until an "*Acknowledge*" signal comes. When this happens, the transmitter switches back to the data transmit condition.

In the receiver, the intermediate ranging mode is selected when eight blocks running come in error or repetition is requested for the same data block four times in succession (an erroneous block and a correct block alternate for the duration of eight cycles).

The "end of transmission" condition arises in the transmitter upon cancellation of the "*Request transmission*" signal from the computer. In this case, the last data block is followed by a block containing the "*End of transmission*" flag. After this block is transmitted and an "*Acknowledge*" is received, data transmission

is discontinued. The terminal is disconnected from the communication link by a signal from the computer or by pressing the "*Data*" key on the control console.

After it receives the "*End of transmission*" flag, the receiver passes this signal on to output, and the "*End of transmission*" light illuminates on the control console of the terminal.

The "monitor" mode is utilized to check the equipment for likely malfunctions. This can be done by running a test transmission at each terminal separately.

CHAPTER 6

PREPARATION OF DATA FOR ENTRY INTO THE COMPUTER

A management information system has to deal with data widely differing in contents and presentation. A good proportion comes in alpha-numeric form. However, a digital computer cannot accept this information unless it is converted to a form convenient to the machine. This stage is called *machine document preparation*, after which data can be transferred into the computer.

Preparation of primary machine documents is the most labour- and time-consuming stage. The rate at which this can be done mainly depends on the operator's speed and is several characters per second. Because of this, input-data preparation devices are grouped and operated off-line, that is independently from the computer, on an asynchronous basis. Within such a group, work can proceed in parallel, on several devices at a time, and basically involves keypunching by the operator, verification of punched data, sorting, and transfer of data for entry into a computer.

Sometimes, input preparation can be performed automatically, including data entry into a computer. In other cases, it may be possible to use a combination of both techniques, with a proportionate decrease in contribution from the human operator.

At present, automatic input preparation is carried out in one of several ways as follows:

1. Source documents are prepared in a form convenient for automatic data reading (dual cards, coding forms, etc.).
2. Machine documents are prepared as by-product of source-document preparation. This can, for example, be done on a conventional typewriter with a tape-punch adjunct.
3. Machine documents are prepared by data loggers installed in production departments, warehouses, and elsewhere.
4. Punched tape can be prepared automatically as data come in over telegraph lines.

The level of automation in input preparation has a direct bearing on the occurrence of errors. Errors are most frequent where input data are prepared by the human operator.

Because errors are inevitable, punched data have to be verified. Most often, verification is done by re-punching the same data and comparing the two sets of punched cards or tape. Experience has shown that re-punching and comparison is a far-more efficient procedure for verification than a visual check.

The number of errors in machine documents is drastically reduced where the responsibility for their occurrence is placed with the originator(s) of source documents.

The choice of a manner in which machine documents are to be prepared is decided by the form and character of source data and the availability of appropriate equipment.

In management information and process control systems, various facilities may be used for input preparation. A sizeable proportion of these facilities are part of the peripherals supplied with each digital computer.

6.1. PUNCHED CARDS AND PUNCHED TAPE

Punched cards. A *punched card* is a rectangular piece of sturdy high-quality paper on which data are recorded in the form of holes. Under USSR State Standard GOST 6198-64, punched cards should measure 187.4 ± 0.1 mm in length, 82.5 ± 0.1 mm in width and $0.15^{+0.015}_{-0.010}$ mm in thickness. As is seen, punched cards are made to fairly stringent dimensional requirements. This is done because the transport mechanism should pick up cards precisely one at a time and pass them without cocking. The card thickness has been chosen to provide ample electrical insulation, stiffness and ease of punching.

The material for punched cards is likewise chosen to be mechanically strong, as each card has to pass through the computer many times. Under GOST 6198-64, punched cards should stand up to a hundred passes through a computer at the rate of 17 per second and fifty passes at the rate of 33 to 42 per second without loss of quality.

Punched cards should be non-conducting for electric current and opaque to light, so that they can be read by an electromechanical or an optical method. Punched cards should be insensitive to variations in ambient temperature and humidity and be stored under conditions unlikely to cause changes in dimensions and quality.

The upper left-hand corner of a punched card is cut to keep the cards right end up in processing. The side of cut is 7 mm and the angle of cut is $45 \pm 2°$.

Data are arranged on each punched card in rows and columns. As a rule, a punched card has 12 rows and 80 or 45 columns (there are cards with other numbers of rows and columns, but they are seldom used).

Any symbol is recorded in a card by punching a hole in a particular position at the intersection of a row and a column. The holes punched in 80-column cards are rectangular (measuring 1.4 mm by 3.2 mm), and those in 45-column cards are round (3.2 mm in diameter).

The columns in an 80-column card are spaced on 2.21 mm centres, and those in a 45-column card, on 3.97 mm centres. The spacing between rows in both types of card is the same and equal to 6.35 mm.

On the face side of a card, nine rows (counting from bottom up) are numbered *0* through *9*; the two topmost rows, *11* and *12*, are unnumbered. The holes in these rows are called zone punches and are used to represent some conventions or to control operation of some mechanisms in the machine. The columns are numbered *1* through *80* from left to right by smaller figures above the rows numbered *0* through *9*. The two sets of numbers aid the operator in checking the recorded data visually.

Numbers are recorded in a card by punching one hole in any given column for each digit. For this purpose, use is made of the code to GOST 10859-64 (see the Appendix). The sequence in which the code symbols are punched in a card is illustrated in Fig. 6.1 and in the Appendix.

Keypunching is the most labour-consumping job in the preparation of machine input data. According to statistics, an average operator turns out one erroneous card in two hundred. All kinds of errors may occur: an extra hole punched, a hole missing where one should be keyed, an extra number keyed in, a number skipped, a wrong number punched, a shift of one column or more. The errors due to the operator account for 94 to 97% of the total, mainly because the operator has not been attentive enough. This is why verification is a mandatory step in machine input preparation.

Most often, punched cards are verified: (a) by a sight check (that is, by sighting through the holes of aligned cards towards

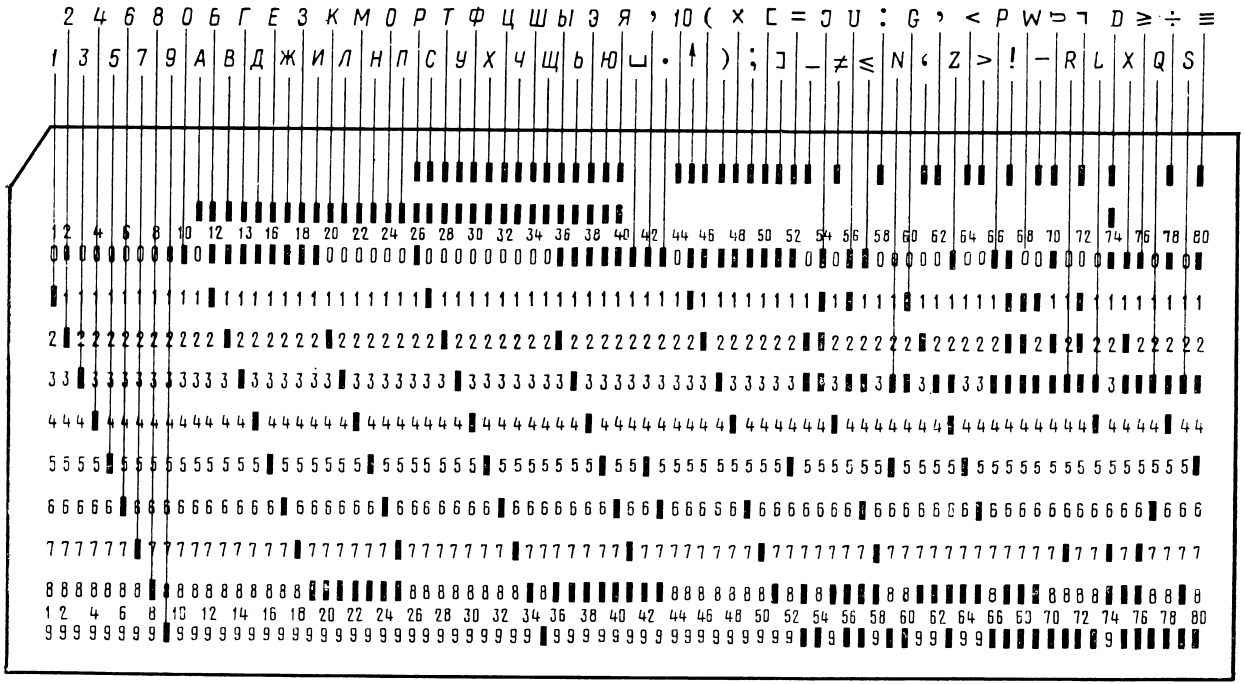

Fig. 6.1. 80-column card for alphanumeric data in the code to GOST 10859-64

a light); (b) by taking batch totals; and (c) on special machines called verifiers.

In a verifier, the cards are fed in turn into a reading station, the operator keys in the second version of the same information and compares with the first. Recorded data are also checked for correctness before a card is fed into the computer.

For a check on input, data are recorded on a punched card in an eight-unit code, that is, one more digit is added. Each column should have an odd number of punches. Punching is done to a specific rule as follows. Each decimal digit (positions *1* through *10*) is assigned one punch in the respective row: the unity in the seventh (most significant) position is represented by two punches in rows *3* and *9*; the unity in the sixth position by a punch in row *11*; the unity in the fifth position by a punch in row *12*; the unity in the fourth position by a punch in row *8* except where the seventh, third and second positions contain zeros and the first position contains a one (0001001), which corresponds to having a punch in row *9*. Unities in the first three (least significant) positions are represented by punches in rows *1* through *7*, except when the seventh, second and first digit positions contain unities and the third digit position contains zero (1000011), which corresponds to having punches simultaneously in rows *2* and *1* (instead of one punch in row *2*). A punch in row zero makes the total punches in a column an odd number.

The Soviet-made ПА80-6 alphameric keypunch uses a different representation for alphabetic data (Fig. 6.2). To record alphabetic information, it punches two holes in each column. Figure 6.3 shows an IBM card in which decimal numbers are recorded by punching one hole in any given column for each digit, and alphabetic information by two holes in each column.

Since each column can store one character of information, one card can store as many characters as there are columns in it. In other words, a 80-column card can store up to 80 characters of information. This arrangement is simple, but the recording density is low.

It is possible to produce denser records, but they are more difficult to write and read. As an example, Fig. 6.4 shows a punched card in which rows *12*, *11*, *0*, *1* and *2* store two blocks of five-unit binary data, and rows, *4*, *5*, *6*, *7*, *8* and *9*, two blocks of six-unit binary data. The card stores a total of 128 numbers. This type of card is utilized in the MINSK-32 digital computer for service purposes.

Punched tape. The material for *punched tape* may be paper or celluloid. Data are stored in the form of holes. Under GOST 1391-51, paper tape comes in two grades: grade A (unoiled) and grade B (oiled). Under GOST 10860-68, punched tape comes in the following sizes: 17.5 mm wide for five tracks or channels (Fig. 6.5*a*), 22.5 mm wide for seven tracks (Fig. 6.5*b*), and

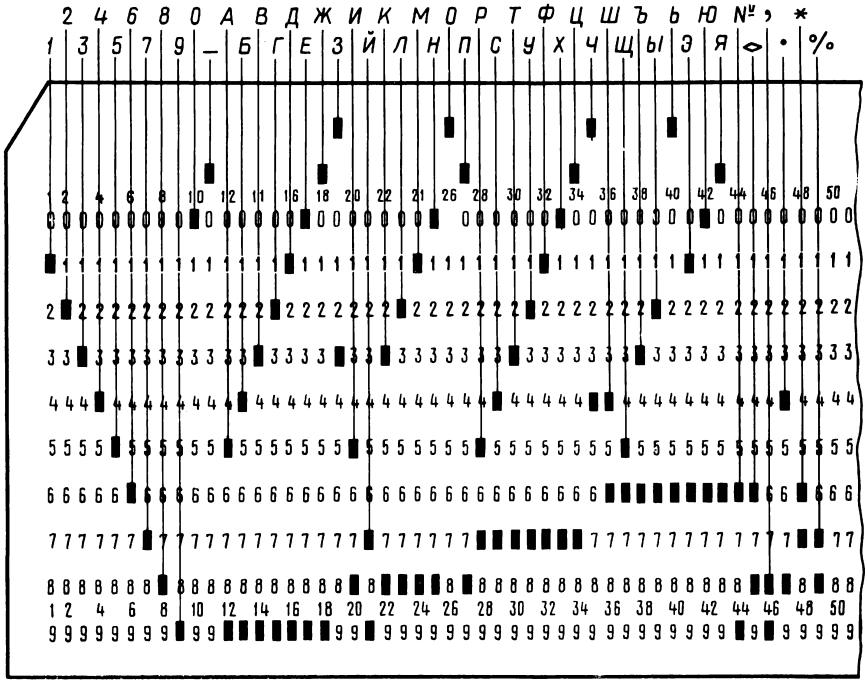

Fig. 6.2. Part of an alphanumeric punched card used by ПА80-6 card punch

25.4 mm wide for eight tracks (Fig. 6.5*c*). The tracks are arranged in parallel along the length of the tape. The spacing between punches, called the *punch pitch*, is made very precise to ensure correct reading (2.5 ± 0.05 mm). The pattern of holes representing a character must be at right angles to the edge of the tape. Hole spacing across the width of the tape is the same as it is along the length of the tape. Punched holes are 1.83 ± 0.05 mm in diameter.

Running horizontally along the length and within 9.96 ± 0.1 mm of the tape's edge are smaller sprocket holes ($1.17^{+0.05}_{0.025}$ mm dia),

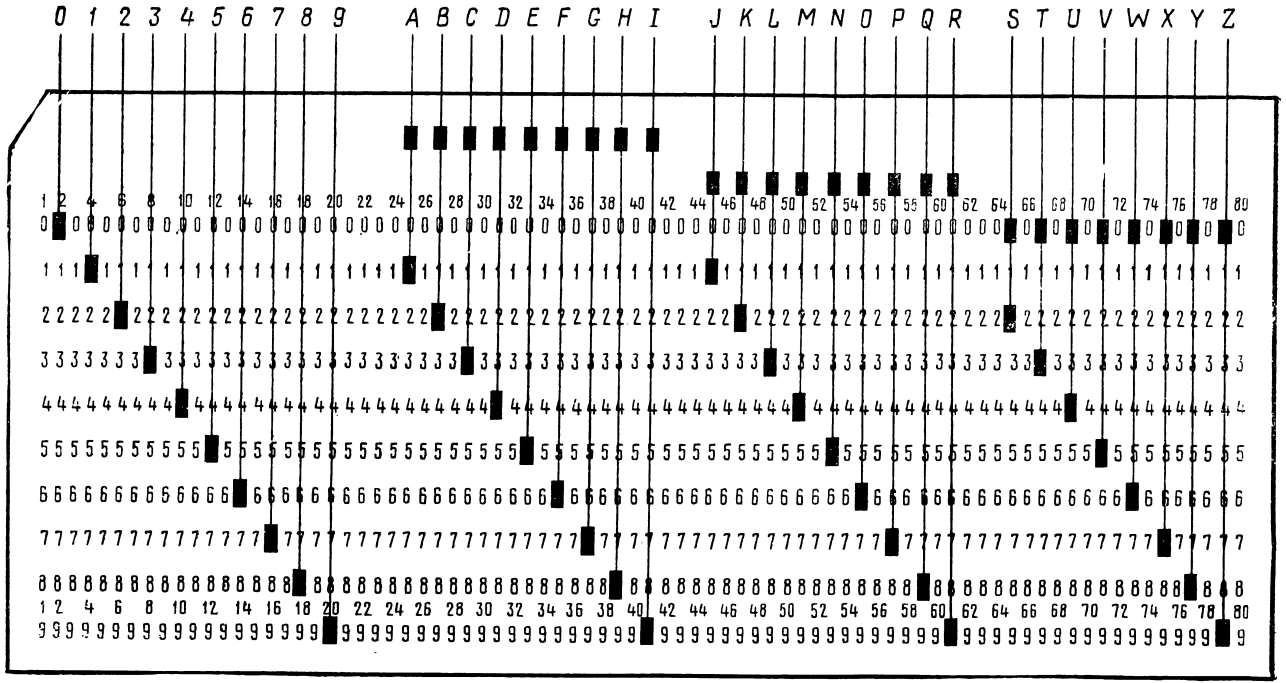

Fig. 6.3. IBM punched card

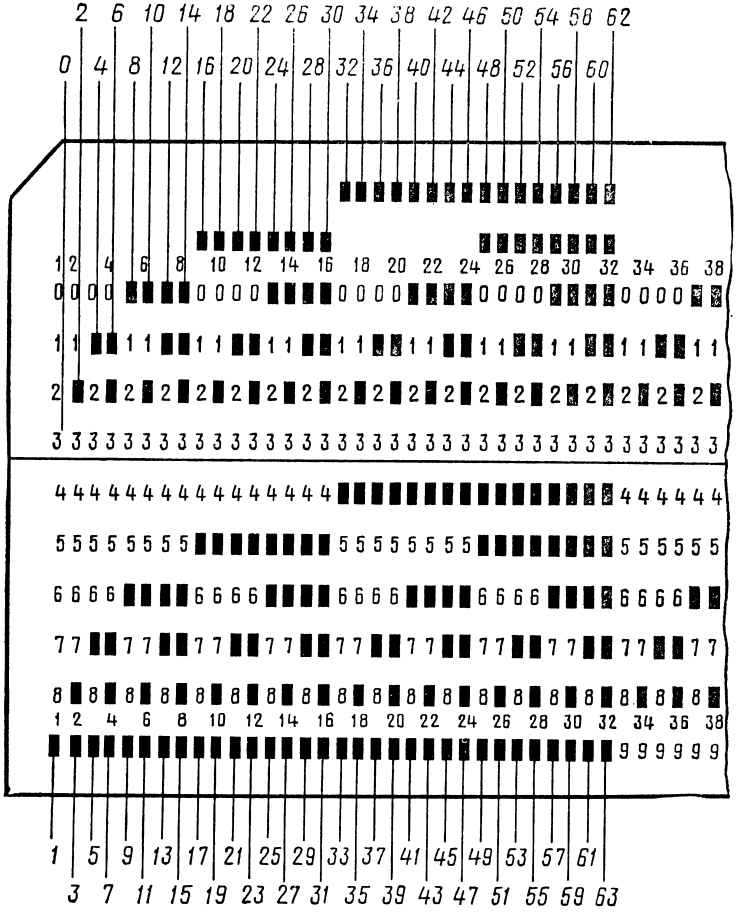

Fig. 6.4. High-density punched card

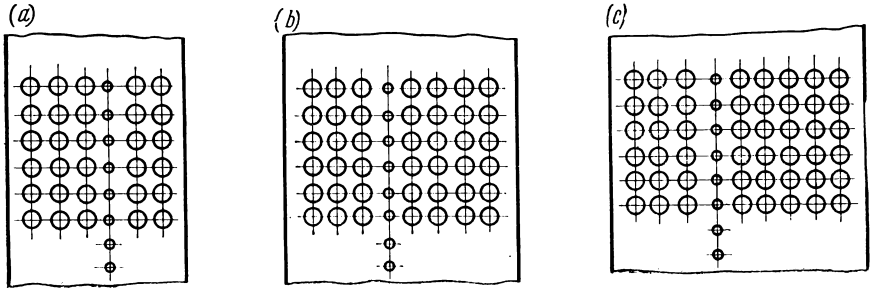

Fig. 6.5. Punched tape

used to feed the tape into the paper-tape reader mechanically. This is a *sprocket* or *feed track*, and the holes are spaced the same distance on centres as the character holes. The material for punched tape is to satisfy the same requirements for quality as one for punched cards. It is to be added that punched tape operates in more adverse conditions as it is continuously bent.

Data stored on punched tape are recorded so that each character is represented by a combination of punches across tape width. On a five-channel tape, each character is coded in a five-unit code, on a seven-channel tape in a seven-unit code, and on an eight-channel tape in an eight-unit code. A punch represents a 1 in the respective code position, and no punch, a 0.

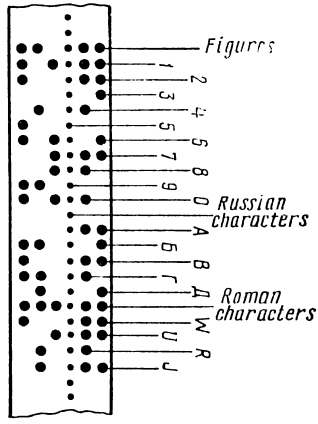

Fig. 6.6. Punched tape with decimal numerals and some letters encoded in international code No. 2

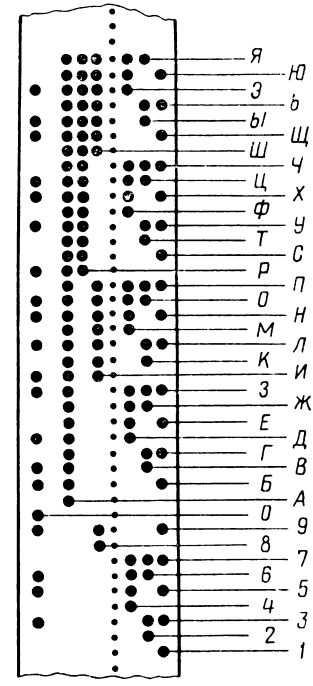

Fig. 6.7. Eight-channel punched tape

The five-channel tape employs the No. 2 international Telegraphic Code widely used in communications and computers. Figure 6.6 shows a segment of a five-channel tape with the decimal figures and some letters recorded in the No. 2 International Code.

As already noted, the seven-channel tape uses a seven-unit

code. Because there is no room for one more digit position that might be used to check the validity of characters (the check bit), the seventh channel tape has found limited use.

The most commonly used type of tape at present is the eight-channel tape. Numeric, alphabetic and special characters are represented in the seven-bit code to GOST 10859-64 in seven channels and the eighth channel is used for an odd parity check. A segment of the eight-channel tape using the eight-bit code is shown in Fig. 6.7. As already noted, the leftmost, or eighth, channel is used for the odd parity check. On its right are the seventh, sixth, fifth and fourth channels and the feed or sprocket row of holes. On the other side of the sprocket track are the third, second and first channels.

There are tapes with other numbers of channels, but they have either fallen out of use or found limited or special application.

Punched cards and punched tape are prepared by punches or perforators. Sometimes, the latter may be part of a larger input-preparation facility.

6.2. CARD AND TAPE PUNCHES

Card punches. These are devices to record data on standard cards in the form of holes or punches. Card punches may be divided into serial and parallel.

In a serial card punch, keys representing certain characters are depressed one at a time, causing consecutive punching until the job is complete. This punching method is applicable to 80-column cards. Apart from an operator, the keypunch may be actuated by a master card or by a computer.

In a parallel card punch, the desired number of characters to be punched in a blank card are first "keyed in"; then, by depressing a release key, the operator causes all the keyed-in characters to be punched in the card simultaneously. This type of card punch is used for 45-column cards and is more elaborate in design than serial punches (they have 540 punch blades, with twelve blades per column). However, parallel card punches allow the errors noticed at the time of keying-in to be corrected prior to punching-in.

Card punches may be used to enter numerical and alpha-numerical data, and they are classed accordingly. The latter are usually part of the input preparation equipment of digital computers.

In the Soviet Union, the types of card punches most widely used (at the time of writing) are the П80-6 numerical serial card punch, the ПА80-2 alphameric serial card punch, and the ПД45-2 parallel card punch.

ПА80-2 *alphameric serial card punch* (Fig. 6.8). It consists essentially of an electric drive to feed and stack cards, a punching station, a card-feed mechanism, a multiple and complete skip

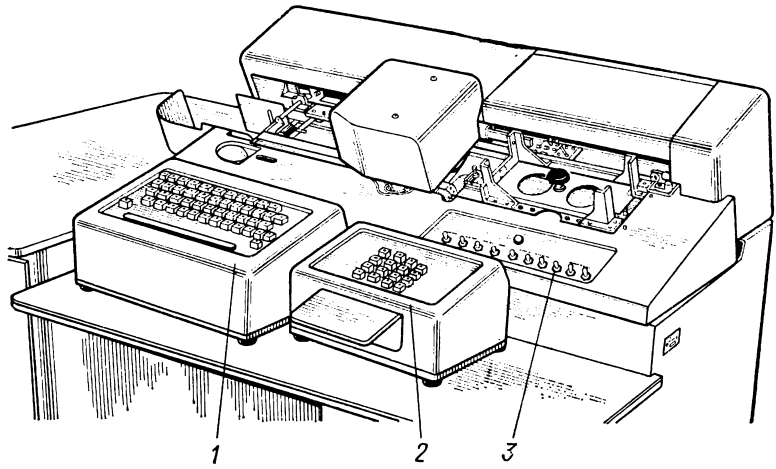

Fig. 6.8. General view of ПА80-2 card punch

mechanism, an alpha-numeric keyboard (*1*), a numeric keyboard (*2*), a control console (*3*), a memory unit, and associated electric switchgear.

A mechanical diagram of the punch is shown in Fig. 6.9.

The electric drive actuates the card bed, hopper and stacker mechanism. It consists of an electric motor, *16*, a worm reduction gear, *17* and *18*, an electromagnetic clutch, *20*, and a meshing gear, *21*. The clutch solenoid is connected in the electric circuit of the drive by sliding contacts, *19*. The electromagnetic clutch couples the motor to the carriage rack, the card bed is returned, a new blank card is picked from the card hopper, and that with the data punched (in the data card) is dropped into the card stacker.

When the sliding contacts energize the clutch, the armature and count gear are attracted to the clutch housing, thereby transmitting rotation from the motor to the meshing gear.

The hopper and stacker mechanism places a new blank card from the card hopper under the aligner arms and moves the completely punched card into the card stacker. In the card hopper, all cards are loaded with their faces up and with the cut on the left. A blank card is placed under the carriage arms by a picker knife, *1*, set up on a slide, *2*.

When the clutch engages, rotation of the meshing gear is transmitted by an idle gear, *6*, to the rack, *9*, and by a meshing assembly, *7* and *8*, to the feed shaft, *5*, another gear *14*, and a centrifugal brake, *15*.

The feed shaft actuates a cam, *4*, and a link, *3*, which drive the picker knife, and also gears *12* and *13* which set in motion the drive rolls, *10* and *11*.

As the slide moves to the left, the picker knife picks the bottom blank card from the deck in the card hopper and pushes it through a slot between the drive rolls which guide it under the carriage arms for punching.

The punching station performs all recording when a card moves and passes through, in accordance with input data. The station comprises a box enclosing twelve punch solenoids (only one is shown in the figure), as many levers and armatures, a die, twelve punch blades (only one is shown in the figure), and a punch bail.

When the punch solenoid, *22*, is energized, the armature releases the lever, *23*, which causes the punch blade, *24*, to move into a rectangular hole in the die, *25*. The die is made up of two members separated by a slot through which the card to be punched is passed. As the solenoid (or several solenoids, as the case may be) is energized, the respective punch blade (or several blades) punch a hole (or several holes) in the column aligned with the die.

After column 80 of a given card passes through the punching station, it is released by the aligner arms and picked up by the stacking rolls to be dropped in the card stacker. At the same time, the aligner is returned to align with the first column position. The centrifugal brake maintains a constant speed for the aligner in the case of multiple or complete skipping.

The advance mechanism advances cards from one column position to the next and is mechanically linked to the punching station. Card advance is controlled by two advance pawls, a release pawl, *33*, and a brake pawl, *29*.

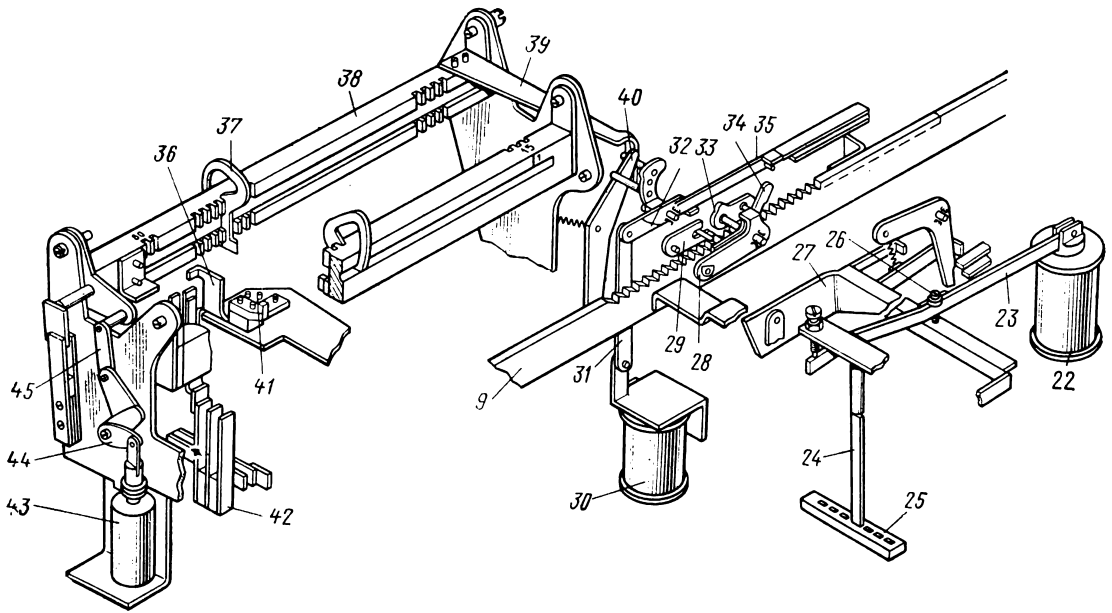

Fig. 6.9. Mechanism of ПА80-2 card punch

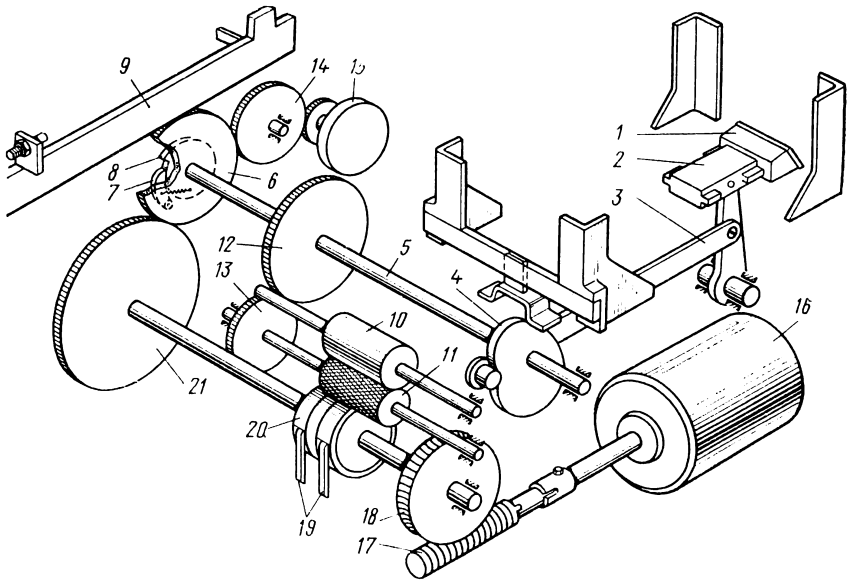

Fig. 6.9. Continued

Initially, the release pawl engages the rack and prevents it from moving; the brake pawl is disengaged.

When the punch solenoid is energized and the lever moves, the adjusting screw, *26*, brings pressure to bear upon the punch bail, *27*, and the latter goes down. As a consequence, the cam, *28*, rotates and actuates the release and brake pawls and lever, *34*. The release pawl disengages, and the brake pawl engages the rack. This change-over takes place before the punch blade comes in contact with the blank card. The brake pawl improves alignment of the card bed and, as a consequence, ensures better alignment of holes in a given column.

When the release pawl disengages the rack, the latter is forced by a spring to move to the right, and the release pawl is positioned above the next tooth of the rack. When the punch solenoid is de-energized, its armature is lifted, the punch bail, cam and lever take up the home position. The brake pawl disengages, and the release pawl engages the rack.

The multiple skip mechanism allows several columns in a given card to be left unpunched. Whenever a skipping function is requested, a solenoid, *30*, is energized, and its armature actuates a

system of levers (*31*, *32* and *35*) to lift the release pawl. Since the brake pawl is likewise disengaged, the spring causes the rack to move left unobstructed; in the meantime, the punching station remains disabled. The rack has a bracket mounting two limit stops, *36* and *41*. As the rack is moved, limit stop *36* travels along the guide comb, *38*. The slots of the comb receive tabs, *37*, set in the positions corresponding to the columns at which the card bed is to be stopped.

Initially, the guide comb is moved out of the way, and limit stop *36* does not catch at tabs *37* as the rack is moved. When contacts *42* make, solenoid *43* is energized, and levers *44* and *45* cause the guide comb to move forward. Now limit stop *36* catches tab *37* and causes the guide comb to move left. Lever *39* brings pressure to bear upon latch *40*, thereby releasing lever *35*. Lever *39* and release pawl *33* take up their home positions, and the card bed comes to a stop. Solenoid *43* is de-energized, and the guide comb takes up its home position.

The punched cards are counted by a mechanical counter. Each time a punched card is dropped into the card stacker, a mechanical contact is closed and a solenoid is energized. As a result, its plunger moves and adds a unity to the previous count.

The ПА80-2 card punch has a numeric and an alphametric keyboard. Each keyboard is physically a separate unit electrically connected to the remaining elements by a cable and connectors.

The alphameric keyboard, *1*, of the ПА80-2 card punch (see Fig. 6.8) had 32 keys (for the letters of Russian alphabet), ten keys for figures, six of which (numbered 4 through 9) are combined with the symbols No, ◇, ",", ".", "*", and "%", a "—" key, and six control keys.

Four of the control keys are labelled and used as follows:

"МП" (multiple skip), to skip any number of columns in a given card without punching in any data;

"ОП" (single skip), to skip one column in a given card unpunched;

"П" (start), to feed a new blank card after the previous card has been completely punched, while the autostart switch is off (when this key is depressed, the card bed is brought back to its home position and the previously punched card is dropped in the card stacker);

"В" (eject), to eject the card from any column position without further punching-in;

The remaining two control keys are unlabelled and are used for the shift function. They are located on the right and left of the long black "single-skip" key.

Where only numerical data are involved, punching can conveniently be done on the second keyboard (at *2* in Fig. 6.8). It has ten numeric keys, two keys to make zone punches in rows *11* and *12*, and four control keys labelled "МП", "ОП", "П" and "В".

The various modes of operation for the punch can be selected with the ten toggle switches arranged on the control panel (at *3* in Fig. 6.8). The functions of these toggle switches are marked on labels attached above the switches. The labels read as follows:

"*Power*", to apply power to the card punch;

"*Autostart*", to switch the punch to automatic operation (feeding blank cards under the punching station and dropping data cards into the card stacker);

"*Multiple skip*", to start the multiple-skip mechanism. With this toggle switch, the mechanism can be disabled temporarily without affecting the set-up of the punch in other respects;

"*Autopunch*", to enable the punch to key in permanent symbols or characters automatically from its memory unit;

"*Register on*", to switch in the memory register and out any of the register for the duration of data punching;

"*Write in register*", to write or erase data into or from the memory unit. When the respective toggle switch is thrown to the ON position, the previously recorded data can be erased from the respective register and new data recorded instead.

The memory unit consists of three relay registers, a line commutator and a patchboard.

The registers store permanent symbols or characters, single and multiple skip signals. Each register has ten digit positions, so that thirty permanent symbols can be stored in all. In turn, each digit position of a register uses twelve relays (as many as there are punching positions or rows on a card). Characters are written into the registers serially, beginning with the first. Data can be written into and erased from each register independently.

The line commutator selects appropriate digit positions of the memory registers. Physically, it consists of an insulating block carrying 80 contact segments and a slide-arm with triple-contact springs connected to the card bed. Each contact segment represents one column in a card.

The patchboard is provided to establish connections between particular column positions and the respective registers in the memory unit. It has six horizontal rows of holes. Each hole in the first, second, fifth and sixth rows which correspond to the respecitve columns of the card are marked with double numbers. The number above a hole designates the column to be used when the machine is to be operated without a shifter. The number on the left of the hole designates the column to be used when the machine is to operate in conjunction with a shifter. The third and fourth rows have three groups of ten holes each, one group for each register. The holes are numbered according to the digit positions of the registers.

The holes in the patchboard receive patch cords which connect appropriate register digit positions to the holes assigned to the card columns where the permanent characters are to be punched from the memory.

Earlier makes of the ПА80-2 card punch have a shifter and three more control keys on the keyboard.

The shifter is intended to delay punching and has six stages (digit positions) with twelve relays in each. The stages of the shifter are arranged in a ring circuit. Three stages are at work at a time. For example, if a symbol is being written into the fifth digit position, the previous record in the sixth position will be erased, and the symbol recorded in the first digit position will be punched in. During the next clock period, if a symbol is being keyed into the sixth digit position of the shifter from the keyboard, the symbol recorded in the second position of the shifter will be punched in, while that in the first digit position will be erased. Thus, punching-in is delayed from keying-in for the duration of four cycles. This arrangement enables any errors made during the keying-in stage to be corrected in the shifter, and a serial card punch operates, as it were, as a parallel punch.

When input preparation is just begun, the first four symbols are only keyed in, but not punched. The first symbol will be punched in only after the fifth has been keyed in, the second after the sixth has been keyed in, etc. Should the operator notice an error in the keyed-in data, he can depress the "ОГ" (single erasure) key to erase the last symbol in the keyed-in data. Should an error be disclosed two or three cycles later, the "ОГ" key should be depressed two or three times, as appropriate, and the correct data keyed in. The four symbols keyed in last can be erased by depressing the "ПГ" (complete erasure) key, and the

four last symbols can be punched in from the shifter by depressing the "C" (shifter) key.

П80-6 *card punch.* This machine is intended to punch in only numerical data on 80-column cards. This can be done either manually by an operator from a source document or automatically by duplicating from a master card. In the former case, the operator can hand-key data in all the 80 columns of a card, skip one or several columns, or skip a card as a whole.

In duplicating from a master card, the latter is placed in and read by a reading unit, appropriate signals are sent via the control panel to the respective solenoids in the punching station, and this punches holes in the respective columns of the blank card.

The П80-6 card punch is simpler in design than the ПА80-2 machine. Some units, such as the hopper and stacker mechanism, punching station and drive, are in fact identical. The П80-6 punch, however, has no alphabetic keyboard, line commutator, or relay memory unit.

The keyboard is a physically separate unit connected to the punch proper by two connectors. The keyboard has ten "figures" keys (labelled *0* through *9*), two keys to make zone punches in rows *11* and *12*, and three control keys. The latter are labelled and used as follows:

"*Start*", to move the card bed to the rightmost position if the autostart switch is turned off;

"*Complete skip*", to actuate the complete skip mechanism in which case some columns in a given card will be skipped without any data punched in (this key is used when data are to be punched in only some of the columns or an error has been disclosed in the keyed-in data).

The machine also has a switch panel mounting four switches labelled "*Power*" (to apply power), "*Autostart*" (to initiate an automatic cycle of operation), "*Multiple skip*" (to skip several columns at a time), and "*Zone punch*" (to punch two holes in the same column, one in row *11* or *12* and the other in one of rows *0* through *9*). In the latter case, the "*Zone punch*" switch is turned on first, and the respective numeric key afterwards. When the "*Zone punch*" switch is turned off, the card is moved on to the next column after a hole is punched in row *11* or *12*.

In the above machines, especially the ПА80-2, the punched-in data are fairly difficult to verify visually. As a way out, a card punch is now ganged up with a printer so that a readable copy of punched-in data is prepared concurrently with punching-in. In

Russian usage, this combination has come to be called as an УПДК (meaning "*printer-card punch*").

Tape punches or perforators. These machines are simpler in design than card punches, mainly because tape is simpler to advance than cards.

As an example, consider the ПЛ-80 machine using paper tape 17.4 mm and 25.4 mm wide. Data are punched in, using a five-,

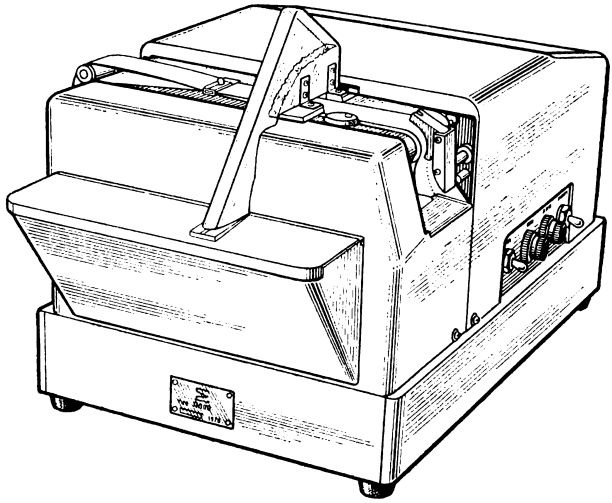

Fig. 6.10. Type ПЛ-80 tape punch

six-, seven- and an eight-unit code. The shape, size and arrangement of holes are in accordance with GOST 10860-68. This is a desk-top machine without a keyboard (Fig. 6.10).

The tape punch consists essentially of a punch action, a code setting assembly, a tape feed, a synchronizer, a drive motor, a torn-tape mechanism, an excessive-tension mechanism, and a chad-disposal assembly.

When the drive motor (at *1* in Fig. 6.11*a*) is energized, pulleys *2* and *4* and drive-belt *3* transmit rotation to an eccentric shaft, *5*, which actuates the mechanisms of the punch machine. On this eccentric shaft is mounted a lobed synchronizer disc, *16*. At the instants when the lobe passes under the cores of the synchronizer coils, *SC-1*, *SC-2* and *SC-3*, pulses are induced in them, utilized as output signals routed into external circuits. The signal from *SC-3* signifies the *start of a cycle*, that from *SC-1* a *readiness* or *tape threading condition*, and that from *SC-2*, *code reception*.

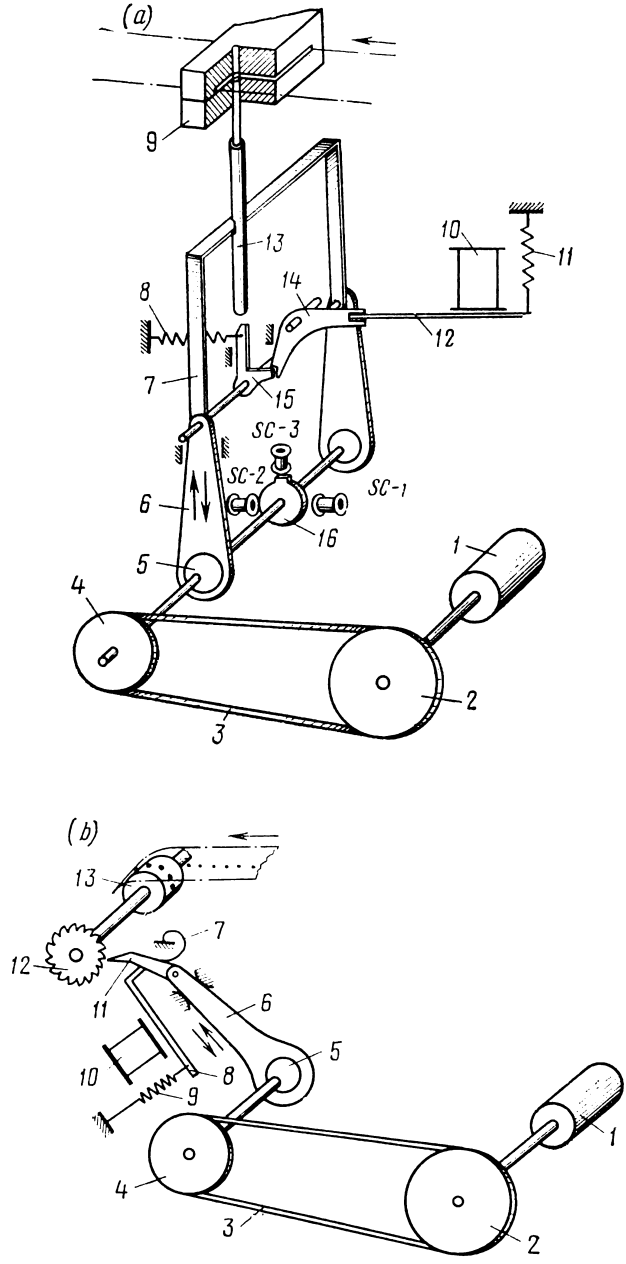

Fig. 6.11. Type ПЛ-80 tape punch:
(*a*) punching station; (b) feed mechanism

When the "*Receive code*" signal is generated, the code and feed-track magnets are energized. The "*Ready*" signal terminates the signals applied to the code and feed-track magnets, and an "*Advance tape*" signal is applied to the tape-feed magnet. In the "*Thread tape*" condition, signals to advance tape and to punch feed holes are only generated.

The machine can operate in any one of two modes, namely punching and idling.

In the punching mode, the "*Receive code*" signal causes appropriate code magnets, *10*, to be energized with tape-punch signals (these signals represent five- or eight-unit code combinations). As a result, the energized magnets (only one magnet is shown in the figure) overcome the opposition of springs *11* and attract their respective armatures, *12*. This causes rotation of the interposer arm, *14*, so that the interposer, *15*, moves against a spring, *8*, to take up a position under the punch blade, *13*. As the eccentric shaft, *5*, rotates, it actuates the punch bail drive link, *6*, and the interposer, *15*, thereby causing the punch blade to move upwards and punch a hole in the tape held stationary in the die, *9*. After the hole has been punched, the punch bail resets the punch blade. Then the "*Advance tape*" signal applied to the tape feed magnet, *10* (see Fig. 6.11*b*), causes it to feed the tape one step forward. When the feed solenoid is energized, the motor, *1*, transmits rotation via pulleys *2* and *4*, drive belt *3* and eccentric shaft *5* to punch bail drive link *6*, which moves upwards and causes pawl *11* to press armature *8* closely to the core of solenoid *10*. Overcoming spring *9*, armature *8* is held attracted by the magnetic force of the solenoid.

When the punch bail drive link moves down, the pawl is acted upon only by the spring (at *7* in Fig. 6.11*b*). When the punch bail drive link reaches the lowermost position, the pawl engages the next tooth of ratchet wheel *12* to turn it one tooth forward as the drive link moves upwards, thereby causing tape feed drum *13* to advance the tape one step forward.

In the idling mode, the tape is not punched.

A different arrangement is utilized in the Perfomom-35 perforator intended to punch data in five-, six-, seven- and eight-bit codes on cards and tape in the form of edge holes. Its maximum punching speed is 33 rows per second.

The perforator comprises a punching station, a tape-feed mechanism, a synchronizer, an end (break)-of-tape detector, a start-

of-card detector, a chad-disposal system, and appropriate controls.

A simplified diagram of the punching station for one digit position is shown in Fig. 6.12. When the motor is energized, its rotation is transmitted by a drive belt to drive shaft *2*. On the shaft is made fast an eccentric, *1*, which causes the bail, *5*, to move to and fro. The bail carries an interposer, *7*, free to oscillate within narrow limits. If a hole is to be punched in a given digit position, the code magnet, *6*, is energized the same instant as the bail starts moving upwards, and attracts the interposer arm, *3*, so that it engages the left-hand end of the interposer. At the same time, the bail moves the right-hand end of the interposer upwards, and the interposer strikes the punch pin, *8*, to make a hole in the tape.

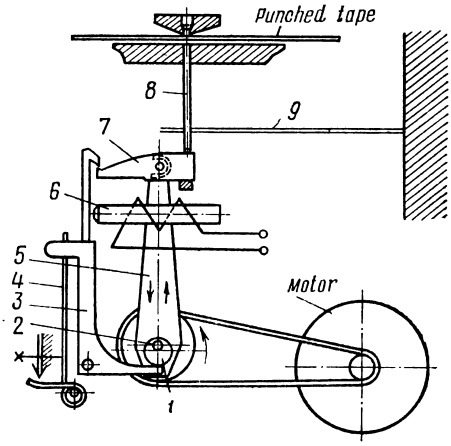

Fig. 6.12. Punching station

After that, the synchronizer de-energizes the solenoid, spring *4* resets interposer arm *3*, and spring *9* resets the punch pin.

If code solenoid *6* is not energized during the upward motion of bail *5*, the interposer arm will not engage interposer *7*, the latter will rock on the pivot without striking punch pin *8*, and no hole will be punched.

The tape-feed mechanism (Fig. 6.13) consists of a step action and a sprocket, *1*, mounted on its shaft and advancing the tape. The shaft, *7*, and a 26-pole armature, *8*, are free to rotate in bearings installed in the housing, *2*, of the step action. The shaft carries a bearing-mounted 26-pole rocking rotor consisting of a cup, *5*, an external yoke, *4*, and a field coil, *6*. The stator, *10*, likewise has 26 poles, and a field coil, *9* of its own. The rotor is coupled to the eccentric of the punching station by a drive link, *3*, which causes the rotor to swing with a double amplitude equal to the pole pitch. If, during a given cycle, the rotor swings in the direction in which the sprocket should move and, as a consequence, the tape should be advanced, field coil *6* sets up a magnetic flux which threads yoke *4* and cup *5* and has its path completed through

armature *8*. The resultant magnetic force applied to armature *8* causes the shaft to rotate one pole pitch. If the rotor is reversed during a given cycle, coil *6* is de-energized, but stationary field coil *9* of stator *10* is energized, so that armature is stopped and prevented from rotating in the reverse direction.

If the tape is to be advanced in the opposite direction, coils *6* and *9* are energized in the reverse sequence.

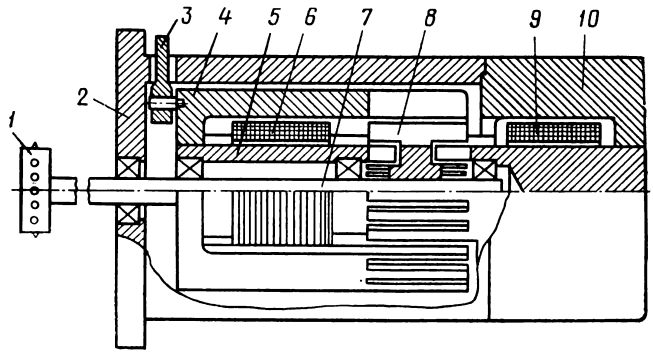

Fig. 6.13. Tape feed

The punching station and the tape-feed mechanism are driven by a single 220-V, 50-Hz, single-phase induction motor operating at 2800 rpm.

The synchronizer (Fig. 6.14*a*) generates timing signals which synchronize input data with the punch and the internal elements of the punch with one another. The synchronizer consists of a lamp, *1*, a double-slit disc, *2*, mounted on an eccentric shaft, and two photocells, *3*. As the shaft rotates, light admitted through the slits of disc *2* reaches the two photocells in a definite sequence, so that two sequences of pulses, *I* and *II*, appear at the outputs of the photocells. Sequence *I* represents an "*Advance*" signal, and sequence *II*, a "*Ready*" signal. The "*Advance*" signal determines the time interval during which the tape-advance coils remain energized and also the instant when the tape-feed mechanism is braked. The "*Advance*" and "*Ready*" signals together determine the time interval during which the code magnets of the punching station are energized.

The "end-of-tape" (torn-tape) detector is a microswitch in which the contacts make when the tape is threaded through the tape guide and break when there is no tape in the guide.

The "start-of-tape" (card) detector is likewise a microswitch which will remain closed until the hole marking the start of tape (or card) reaches it. At that instant the microswitch opens.

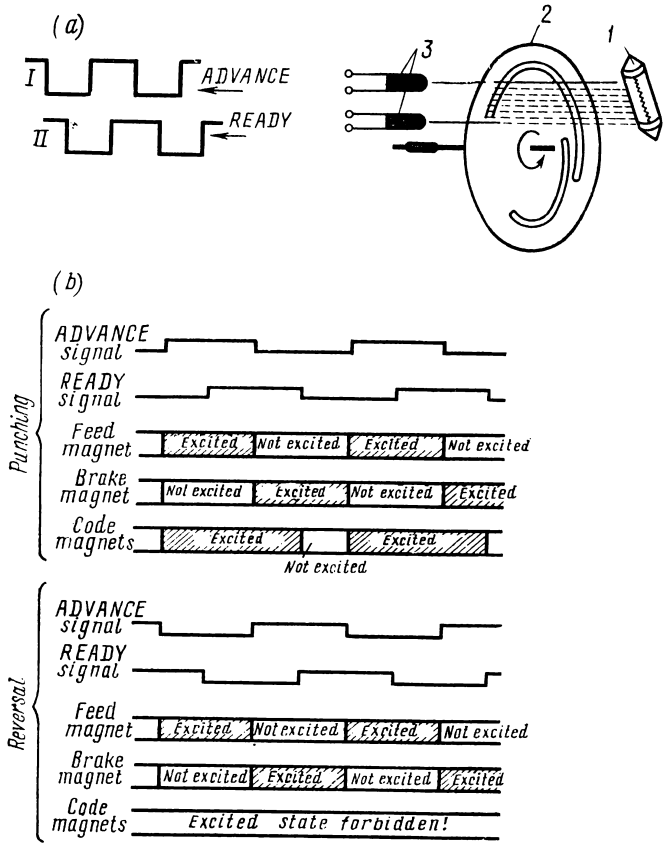

Fig. 6.14. *(a)* Signal generator; *(b)* timing diagrams

The chad-disposal system removes chads as they are produced in punching. It is essentially an impeller driven by the drive belt of the perforator.

The controls include a key marked with an arrow, a key labelled *M*, a key labelled *B*, three sunk knobs (*1*, *2* and *3*), and an indicator lamp (*4*) which illuminates when the perforator is energized (Fig. 6.15). The arrow-marked key is used to reverse the direction of tape (or card) advance by hand. When the *M* key is

depressed, holes will be punched in all tracks (this is a test condition known as "*Contiguous punching*"). The *B* key is depressed when the operator wishes to feed the tape forward. Now only feed or sprocket holes will be punched on tape, while a card will be advanced with no punching at all. Knobs *1* and *3* set the spacing between the side stops according to the width of the tape in use. Knob *2* has a numbered dial to set back-spacing up to 25 lines.

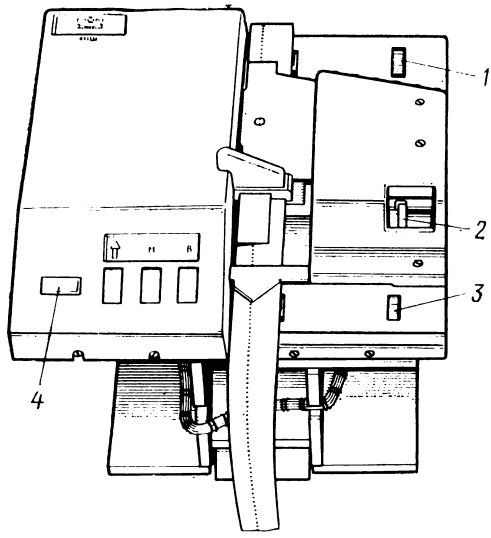

Fig. 6.15. General view of a tape/card punch

The electronic control unit is built into a separate case.

Tape punches, or perforators, are usually employed as part of input preparation equipment or as output perforators for computers.

6.3. CARD AND TAPE READERS

In automatic control and management information systems, card and tape readers are widely used in a variety of applications: in perforators, input preparation equipment, data loggers, computer entry facilities, data communication, etc. Card and tape readers convert the patterns of holes into appropriate sequences of signals.

According to the technique used to sense holes in cards or tape, readers may be classed into mechanical, pneumatic, electrostatic and photoelectric.

Mechanical readers generally sense holes by driving a contacting element towards the punched card or tape. If the contacting element passes through a hole, electric contacts close to complete the signal circuit, and an appropriate signal appears at the reader output. The contacting elements may be brushes, prods, needles or pins. Mechanical readers are simple in design, but can operate at low reading rates because of the inertia of their mechanical contacts.

Pneumatic reading consists in the following. An air stream is allowed to pass through holes in the card or tape and change the state of a suitable transducer with the result that a signal appears at the reader output. The transducers may be temperature-sensitive resistors, switch-actuating membranes, or piezoelectric crystals.

An *air-jet reader* has a system of tubes arranged coaxially. The punched card (or tape) is advanced between the ends of the jet tubes. When a hole in the card or tape appears between the tube ends, an air stream flows, and the pressure in the tubes drops. Changes of pressure are converted by pressure transducers into electric signals.

Among the advantages of air-operated readers are the high rate of data reading, simple design, and reliability of operation under adverse service conditions.

Electrostatic reading utilizes changes in the capacitance of the reader's capacitors at the instant when a hole in the card or tape passes between the plates (these changes are then converted to electric signals).

The sensing elements in electrostatic readers are miniature capacitors whose plates are arranged on either side of the card or tape passed through, or capacitors made up of miniature plates on one side of the card or tape and a common plate on the other side.

Each miniature capacitor is connected in an arm of a bridge energized with alternating current at a frequency of several hundred kilohertz. The bridge is balanced with a variable capacitor. Across the bridge is connected an amplifier tuned to the supply frequency. The amplifier output is coupled to a detector which produces a rectangular pulse at the instant when a hole in the card or tape passes by the respective miniature capacitor.

Among the advantages of electrostatic readers are high reading rate, simple design, and high reliability.

Photoelectric reading is based on changes in the resistance of a photocell under the action of incident light. These changes in resistance are then converted to electric signals.

A photoelectric reader consists of a light source, an optical system to focus light from the source into a narrow beam and to direct it onto the photocell head past which a card or tape is moved.

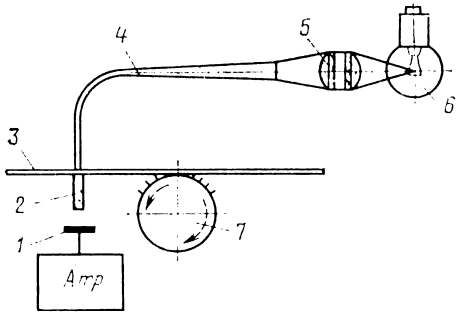

Fig. 6.16. Photoelectric reader

Various lamps may be used as light sources, while the photocells may utilize photoresistors, photodiodes and phototransistors.

At present, photoelectric readers are the most widely used devices. Their major advantages are high reading speed and a relatively simple design.

The Soviet-developed unified system of electronic computers (designated "ES EVM" for short) uses the Readmom-40 photoelectric reader. It can read punched tape and edge-punched cards, *3*, using a five-, six-, seven- or eight-unit code. Each channel of the reader incorporates optical systems (at *5* and *2* in Fig. 6.16), a lamp, *6*, a light pipe, *4*, a photocell, *1*, a message signal preamplifier, *Amp*, an electric step motor driving a sprocket, *7*, and a latch mechanism.

The feed mechanism is very simple owing to the use of a step motor and also because the tape-feed sprocket is made fast to the motor shaft. The motor is braked by the latch.

The flexible light pipes have allowed the reader elements to be arranged in a more compact way.

The Readmom-40 can operate in one of two modes: start-stop reading (reading a line at a time) at speeds from 0 to 40 lines per second, and continuous reading (reading data in blocks) at a speed of about 150 lines per second. The reader will normally operate in the former mode, while the latter is intended for cases where data are to be located rapidly. Tape or cards can be passed in either direction.

Since readers normally operate in conjunction with other devices, they are to satisfy the same requirements for reliability, simplicity of design and reading speed.

Where data are read for entry into a computer, it is sought to achieve the highest reading speed practicable. An example of a high-speed reader is the FS-1501. It is mainly used to read data from punched tape for entry into a digital computer (see Chapter 7).

6.4. ELECTRIC TYPEWRITERS

Electric typewriters are widely used as adjuncts to computers for the preparation, read-in and read-out of data. As their name implies, electric typewriters, like their conventional counterparts,

Fig. 6.17. Typewriter

put out data in type-written form in addition to their input preparation and entry functions. In appearance, too, they look like conventional typewriters (Fig. 6.17).

When used for input preparation and entry, a typewriter is basically intended to generate code signals representing the symbols to be read into a computer.

A standard keyboard-entry typewriter (Fig. 6.18) incorporates a keyboard with a set of keys, *1*, key levers, *3*, with teeth, *12*, and opposing springs, *2*; a printing action composed of a set of type bars *4* carrying type characters, *5*, an ink ribbon, *6*, to transfer an impression of a character onto paper, *7*, a power roll, *8*, a pinch roller, *9*, and a solenoid *10*; and an encoder, *11*.

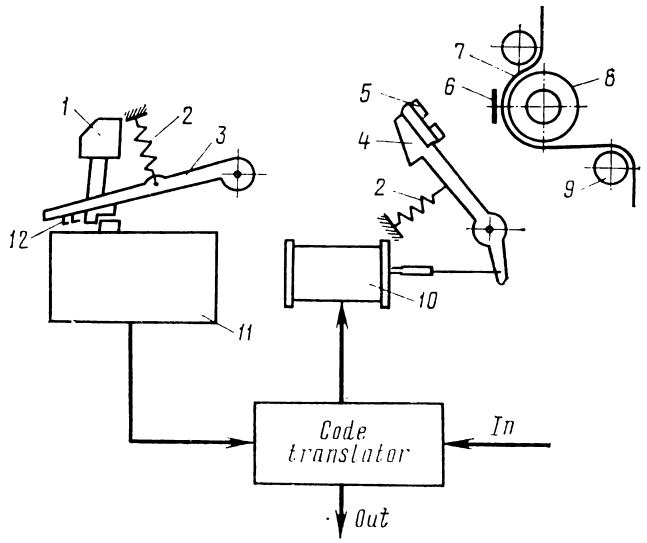

Fig. 6.18. Electric typewriter action (simplified)

The keyboard has information (or message) keys with which data can be put in symbol by symbol manually, and special (or service) keys. The number of information keys is chosen to contain all the symbols needed for an input code. As a rule, each key bears two symbols. Those at the top are the upper-case characters, and those at the bottom, the lower-case characters.

At present, computer-entry electric typewriters of Soviet manufacture (for the Russian language) have forty-six keys (as stipulated by USSR State Standard GOST 14289-69). The upper case contains figures, Russian letters and some special characters. The seventh digit position of all code combinations for these symbols is occupied by a 0 (see the Appendix). The lower case contains all Roman letters and the remaining characters. The seventh digit position of the code combinations for these symbols is occupied by a 1. This arrangement of symbols into an upper

and a lower case simplifies data encoding. The message keys are arranged in four rows. The top row contains figures and some non-alphabetic characters; the other rows contain the letters and the remaining non-alphabetic characters.

The special (or service) keys are used for control of the typewriter. Their functions will be clear from reference to Fig. 6.19. Under the keyboard are six keys with which to select a particular function for the electric typewriter.

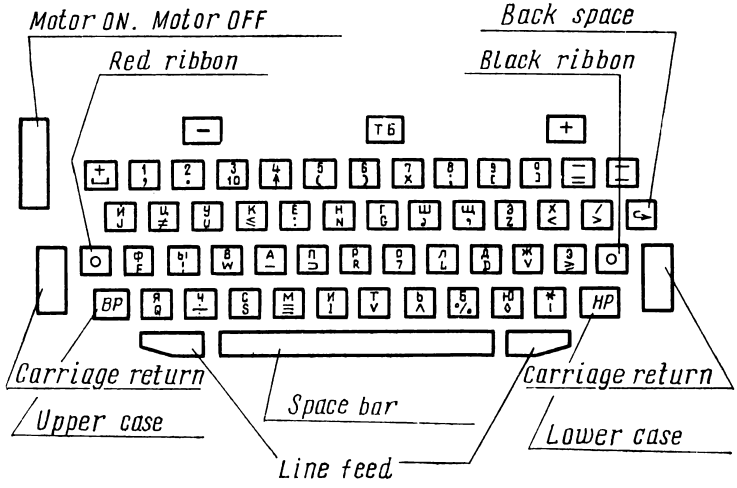

Fig. 6.19. Typewriter keyboard

The message keys are attached to key levers each of which has a projection and code teeth *12* (see Fig. 6.18). The code teeth on each key lever are arranged to represent a particular code combination. Then, pressing a key will actuate only one particular set of coding bars. In turn, the latter will switch the respective set of contacts in the encoder (Fig. 6.20), so that its output delivers a character in an eight-unit parallel code. A unity in a code combination is represented by a current pulse on the output wire (the respective contact is closed), while a zero is represented by no current (the contact is open).

The output wires (*I* through *VI*), assume a 1 state (that is, a unity is formed in each of the first six digit positions) in accordance with the first six positions of the code combination assigned to the symbol whose key has been pressed. To enhance reliability, normally closed contacts are used. Therefore, when a key is pres-

sed to produce a particular character, all contacts break except those associated with the character being transmitted.

The seventh digit position is intended to show whether a particular symbol is from the upper or lower case. It is selected by twin contacts labelled *UC* (for "upper case") and *LC* (for "lower case"). When the keyboard is shifted to the upper case, the *UC* contact opens, and the *LC* contact closes. One of contacts K_{81} or K_{82} also opens. No current pulse appears on output wire *VII*, to represent a 0. When the typewriter is shifted to the lower case, the *LC* contact opens, the *UC* contact closes, and so does contact K_7, and a current pulse appears on output wire *VII* to represent a 1 state.

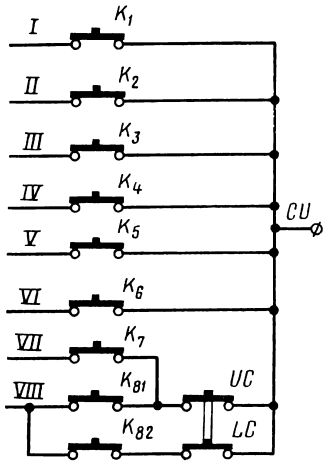

Fig. 6.20. Contact set of a coder

With the keyboard shifted to the upper case, a current pulse (a 1) appears in the eighth digit position (output wire *VIII*) due to closure of contact K_{82} (while contact K_{81} must be open, lest the eighth digit position should affect the seventh digit position). When lower-case characters are keyed in, a 1 is produced in the eighth digit position by closure of contact K_{81}. Contact K_7 generates a 0 by opening the circuits in the combinations existing in both cases. These combinations represent service functions such as "*Carriage return*", "*Red ribbon*", etc. (see Fig. 6.19).

From the encoder, the coded signals go to a code translator which is usually arranged externally to the typewriter (as a rule, it is located within the control console). The code translator converts the coded signals of the typewriter into the code used by a card or tape punch or a computer.

The printing action produces a type-written copy of the data handled, in accordance with the code signals coming from either the keyboard or an extraneous source. The design of this unit has a direct bearing on the reliability and speed of printing. While in operation from the keyboard the printing speed is relatively low and mainly depends on the operator's skill, in data printout the speed is solely decided by that of the printing action of the typewriter. This is why designers usually seek to make the printing action as fast as practicable.

Most often, printing is done by causing each type bar to strike an ink ribbon or carbon paper in order to transfer the desired character onto paper.

Mechanical printing mechanisms print by means of either type bars or print wheels. A type-bar printing mechanism is shown in Fig. 6.18. All characters, *5*, are arranged singly, in twos or more (according to the number of cases used) on type bars, *4*, each actuated by a solenoid of its own, *10*. When a particular solenoid is energized by a signal from the keyboard, its plunger is attracted, the associated type bar is turned so that the character it

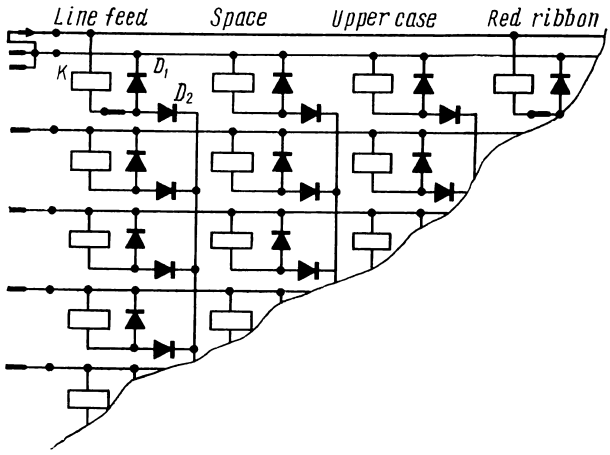

Fig. 6.21. Part of the print-magnet control circuit

mounts strikes the ink ribbon, *6*, and an impression of the character is left on the paper, *7*. Then the platen, *8*, on which the paper is wound, is shifted one space to the left, and the next character is printed in a similar way. In shifting between the upper and lower cases, the platen is moved vertically. Special functions, such as slew (or vertical skip), horizontal skip, carriage return, selection of red or black ink ribbon, etc. are selected by appropriate solenoids or their equivalents.

Figure 6.21 shows a partial schematic of the circuit which controls printing solenoids in the Consul typewriter. A complete control circuit is an 8-by-8 matrix. The top row has eight solenoids intended to select service functions; the remaining rows contain the solenoids that operate the 46 type bars. A particular solenoid is selected by a decoder located externally to the type-

writer. The selected solenoid is energized by strong current pulses applied over the respective horizontal and vertical wires. Isolation between the solenoids is provided by diodes D_2, while diodes D_1 connected in parallel with the solenoids in the reverse (non-conducting) direction cancel out any post-action transients.

When this typewriter is used as an output printer, printing is done over an open loop, as it were, because there is no way to ascertain whether the output characters are printed correctly. Meanwhile, the various devices interposed between the computer output and paper (the code translator, register, solenoid-control circuit, solenoids, type bars) may each contribute specific errors corrupting the printed message. This is why it is desirable and, indeed, important to compare the data printed with those put out by the computer proper.

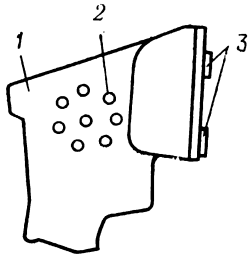

Fig. 6.22. Code holes in a type bar

This is done as follows.

In Facit typewriters, the printed text is checked character by character. To this end, each type bar, *1*, mounting a character, *3*, has eight code holes, *2* (Fig. 6.22). When the character comes in contact with paper, the type bar is illuminated by light from one side. On the other side, there are eight photocells, each positioned opposite a particular code hole. The code holes on each type bar are arranged into a specific pattern. Therefore, when a particular character is printed, the photocells generate the respective code combination which is then compared with the symbol generated by the computer.

Six holes on each type bar would be enough to code all keys (that is, all type bars), as this would produce 2^6 six-unit binary combinations. The seventh hole is utilized for an odd parity check by adding a one to the combination to make an odd total count of unities. The eighth hole in the centre is used for gating.

Type-bar printing mechanisms use a fairly complicated system of levers and arms. In some typewriters, this inconvenience has been avoided by the introduction of a spherical type-head (Fig. 6.23). The characters are arranged on a sphere which is free to rotate in a vertical and a horizontal plane.

An on-the-fly type-wheel printing mechanism is shown in Fig. 6.24. It has a print wheel, *1*, on whose circumference are arranged the characters, a printing hammer, *4*, an opposing spring, *5*, and a solenoid, *6*. The ink ribbon, *2*, and paper *3*, are

threaded between the print wheel and the printing hammer. The print wheel is continuously spinned by an electric motor, and the printing hammer is timed to strike the paper against the ribbon and wheel as the desired character passes. The timing pulses are generated by an electromagnetic sensor. During each revolution of the print wheel, the sensor generates as many timing pulses as there are characters on the print wheel, but the printing hammer is actuated only by those gated out by the decoder. Thus, only one character is printed during each revolution of the print wheel.

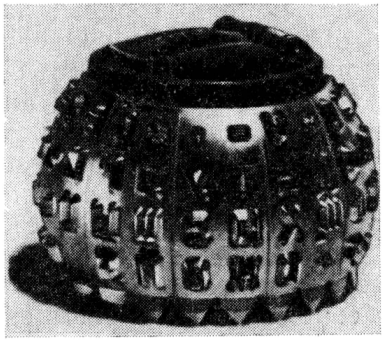

Fig. 6.23. Spherical type-head

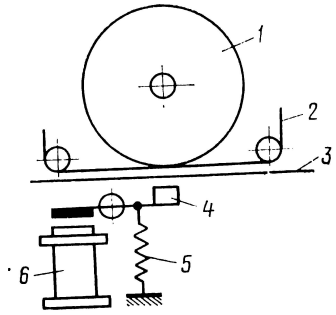

Fig. 6.24. On-the-fly printing action

The number of symbols that can be printed, that is, the number of characters on a print wheel depends on the code used and may be as great as 128 (a wheel is then said to have 128 character or printing positions).

In the Soviet Union, the on-the-fly printing mechanism is used in АПМ-2М and АПМ-3М machines.

The printing speed of type-bar and single-print-wheel on-the-fly machines does not usually exceed 10 characters per second.

It may considerably be increased through the use of several print wheels instead of one. Then each print wheel is arranged to have each character to be printed, and the total number of wheels is made equal to the number of printed positions in a line. Such an arrangement can print an entire line at one time, that is, during one revolution of the print wheels, and the mechanisms are appropriately called line-a-time (or simply, line) printers. In the Soviet Union, they are used in the type АЦПУ-128 output alphanumeric printers.

6.5. INPUT PREPARATION EQUIPMENT

Punched-card input preparation equipment. This class of equipment transfers alphanumeric data from primary documents onto punched cards and produces a type-written copy at the same time. A typical equipment will include an electric typewriter, *ET*, a card punch, *P*, a reader, a physically self-contained control panel, *CP*, and a control cabinet housing the typewriter register, *TReg*, code translators, *CT*, a card punch register, *CPReg*, a punch-magnet control unit, *PMCU*, a typewriter-magnet control unit, *TMCU*, an engineer's console, *EC*, and a control circuit, *CC* (Fig. 6.25).

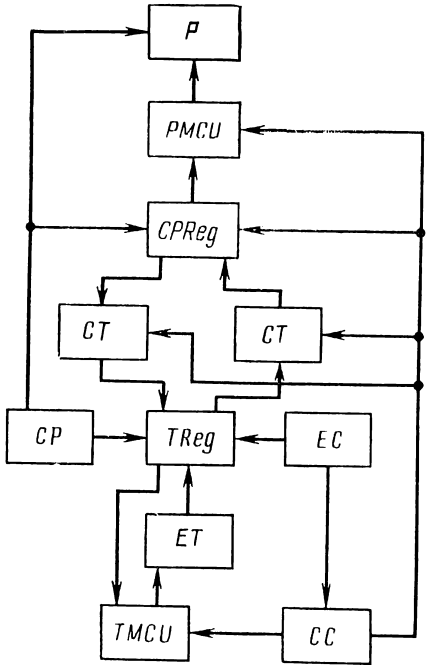

Fig. 6.25. Block diagram of a punched-card preparation device

The operator reads a source document character by character and presses the respective keys on the typewriter. This action closes a signalling contact, and the control circuit generates appropriate control signals, first one to clear the typewriter register, then a "*Keyboard entry*" signal. The latter allows a number to be transferred from the encoder to the typewriter register. Then the code signals go to the code translator where the typewriter code is converted into a punched-card code and entered in the twelve-bit card-punch register.

From the card-punch register, the signals are applied to the power amplifiers of the punch-magnet control unit which routes them to control the card punch. The punched-card code stored in the card-punch register is likewise fed to the code translators for conversion to the typewriter code, which is then stored in the typewriter register. As a check on the validity of each punched character, the contents of the typewriter register must be printed out on paper. To this end, the signals from the typewriter register are routed via a decoder to the typewriter-magnet

control unit so that they can be printed out when a "*Print*" signal comes.

This mode of operation is called serial *keypunching-interpreting* because data are keyed into a card and the respective characters are printed on paper the same instant as the keys on the keyboard are depressed. Arrangements are provided for the selection of other modes of operation. This is done with the aid of function select buttons which are designated (in Soviet practice) as follows:

(1) "*Delay*" — in this mode a character is punched one clock period after it has been printed. As a key is depressed, the respective character is imaged on paper, and the previously type character is punched. To print the last character in this mode, the "*Delay*" key should be released, and any message key should be pressed on the typewriter keyboard. In this mode, any error in the keyed-in symbol may be corrected before the character is punched.

(2) "*Key-in*" — this mode is used when the keyboard does not have the desired character, and the respective combination has first to be keyed in by depressing the appropriate keys on the keyboard and stored as a whole in the card-punch register. Then the total combination (composite character) is punched in the card simultaneously as the "*Key-in*" key is released. On paper, the printed characters appear as a column.

(3) "*Stunt*" — this mode is selected in order to punch such common functions as the selection of red or black ink ribbon carriage return, line feed, etc.

(4) "*Duplicate*" — this mode is used to copy punched data from a master card placed on the card bed of the card punch into one or more detail cards. Unpunched space may be utilized for printed characters keyed in from the typewriter. At the same time, the data being copied from the master card are printed out on paper.

(5) This key is left vacant.

(6) "*Punch disable*" — in this mode the punch is disabled.

The control panel has eight keys labelled "*Clear*", "*Cancel*", "*Skip*", "*Eject*", "*On*", "*Off*", "*Power on*" and "*Power off*".

The "*Power on*" key applies and the "*Power off*" key removes power to and from the equipment via a magnetic contactor.

The "*On*" key energizes and the "*Off*" key de-energizes all power units of the equipment operating on a voltage other than taken from the mains.

The "*Clear*" key is pressed before operating the equipment. This clears (sets to zero) all registers of the input preparation equipment.

Should any punch blade (or blades) stick in the die(s) of the card punch, a relay will operate to remove power after a predetermined delay; when this happens, a "*Protection*" lamp illuminates to confirm that power has been removed deliberately. To restore power, that is, to turn off the "*Protection*" lamp, it is necessary to press the "*Cancel*" key. This will de-energize the relay and restore power.

Pressure on the "*Eject*" key will cause the card to be ejected from any punching position. Pressure on the "*Skip*" key causes the card to move on to the next column.

When depressed, the "*Eject*" and "*Skip*" keys complete the same circuits as the like buttons on the punch panel. In some cases, however, the card punch incorporated in an integrated input preparation equipment may have no control panel of its own. If so, the respective functions can be selected with the "*Eject*" and "*Skip*" keys.

The engineer's panel built into the control cabinet is provided so that the input preparation equipment can be tested and aligned. The engineer's panel mounts:

the "*Start*" button, to start the equipment for the purpose of testing and alignment;

the "*Isolate*" toggle switch, to isolate the equipment from the typewriter keyboard so that any operations can be initiated from the engineer's panel;

the "*Parity*" toggle switch, to disable the parity-check circuit of the card-punch register;

the "*Auto. counter*" toggle switch with which the typewriter register can be instructed to operate as a counter;

the "*Typewriter register coding*" toggle switch, to set various codes for the purpose of alignment and testing.

Also, the engineer's panel has several indicator (or signal) lamps.

Punched-tape input preparation equipment. This equipment transfers alphanumeric data onto tape and produces a typewritten copy of the material at the same time.

A standard punched-tape input preparation equipment includes a tape punch or perforator, *TP*, an electric typewriter, *ET*, and a control console.

Consider the operation of a punched-tape input preparation equipment, using as an example the Brest-1 shown in block diagram form in Fig. 6.26. This system uses a ПЛ-80 tape punch and a Consul typewriter.

In the *input-preparation mode*, pressure of a key in the typewriter causes its encoder, *TWC*, to generate a code combination which is placed in the typewriter register, *TWReg*. From the re-

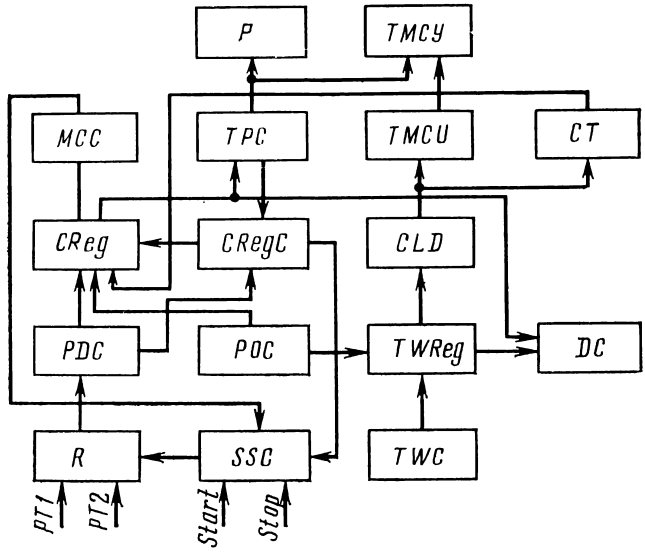

Fig. 6.26. Block diagram of a punched-tape preparation device

gister, the code combinations are fed to a control-level decoder, *CLD*, which decodes the first six digit positions, as the seventh position only identifies the character case (upper, lower, figures) to be used for printing. The three least significant digits and the three most significant digits are decoded separately. From the decoder, the signals are applied to a code translator, *CT*, and to the typewriter-magnet control unit, *TMCU*. As a result, appropriate magnets are energized, and the character of the pressed key is imaged on paper.

A "*CT sample*" signal causes the code translator to transfer its contents to the character register, *CReg*, operated by a control circuit of its own, *CRegC*. When the next key is pressed on the typewriter, the typewriter cycle is repeated and the character

register during the previous cycle is punched on tape. To this end, the signals from the character register are applied via the tape-punch control circuit, *TPC*, to actuate appropriate punch blades in the tape punch. When the punching operation is complete, the code translator is sampled again, and a new code combination is written into the character register. The contents of the character register are also fed to the display circuit, *DC*.

Apart from input preparation, the other modes provided in the equipment include: verification, duplication, verification/duplication, key verification, printout, printout verification.

In the *verification mode*, the equipment compares the data punched on two tapes. For this purpose, a reader, *R*, and a photo-diode circuit, *PDC*, enter the signal representing the coded symbol on the first tape, *PT*1, into the character register which then accepts a similar signal representing the coded symbol in the same line on the other tape, *PT*2. If the two signals match, the character register is cleared to zero. Otherwise, a match signal generator, *MSG*, generates a signal which causes the start-stop control circuit, *SSC*, to stop the reader and actuate a light and an aural signal.

In the *duplication mode*, a duplicate punched tape is prepared. For this purpose, a blank tape is inserted in the tape punch, and a master tape is passed through the reader which senses holes and causes the photo-diode circuit to generate appropriate signals entered in the character register. From the latter, the signals are applied to the tape-punch control circuit, *TPC*, which energizes the respective code and sprocket-hole magnets. The magnets operate, and a line of data is punched into the blank tape. After that, the duplicate tape is advanced to the next punching position, and the character register is cleared to zero in readiness to accept the next code combination.

The *verification/duplication mode* is used to compare two duplicate tapes and, if they match, to punch a third one. In fact, this mode is a combination of the first two.

In the *key verification mode*, a punched tape is verified by keying in the same data again on a typewriter keyboard. The coded signal of the character read from the tape being verified is compared in the character register with the coded signal representing the pressed key on the keyboard. If the two signals match, the match-signal generator supplies a signal causing the character to be transferred from the master tape into the character register, and from the latter into the duplicate tape.

In the *printout mode*, recorded data are read from a punched tape and printed out by a typewriter on paper. If necessary, a duplicate tape can be prepared by an associated tape punch at the same time. Control in this mode is effected by a printout control circuit, *POC*.

The *verification/printout mode* enables data punched on two tapes to be compared and, if the two match, a listing to be printed on paper. This is a combination of verification and printout.

The above modes can be selected with the six "mode select" buttons on the typewriter.

The operator's console is located on the same desk next to the typewriter, while the control panel occupies the top part of the control cabinet. They provide means for controlling and monitoring the operation of the input preparation equipment.

6.6. AUTOMATIC INPUT PREPARATION

As already noted, card and tape punching is a labour-consuming operation likely to produce a considerable number of errors in machine documents. It is, therefore, only too natural that designers have always wished to make this process automatic. Among other things, this goal may be achieved through the use of automatic readers and special documents combining the functions of source and machine documents.

Automatic readers convert data read from source documents into a form presentable to a digital computer. In such cases, no documents need to satisfy any special requirements for presentation or the type used. Unfortunately, existing automatic readers are bulky and slow-working and have not yet found wide use, although work on their improvement is going on.

Special forms of documents combining the functions of source and machine documents are far more elaborate, but the data they carry can be read more easily. Examples of such documents are the dual card and coding forms.

Like ordinary punched cards, dual cards are fabricated from sturdy, high-quality paper and have the same dimensions and tolerances. Dual cards may have overprinted forms which are prescored manually, such as shown in the mark-sensed card of Fig. 6.27. In a mark-sensed card, entries are made in soft-lead pencil and given a particular shape and size. For this purpose a choice of pre-print contours is provided for the entries. Each mark occupies a field equal to three columns in an 80-column card

and represents a digit position in a code combination. A mark-sensed card may be filled out with no more than 27 numerals.

Filled-out mark-sensed cards are then processed by a read-punch which punches holes in the places occupied by the marks.

Consider the operation of a card read-punch, using the ПС-80 device as an example.

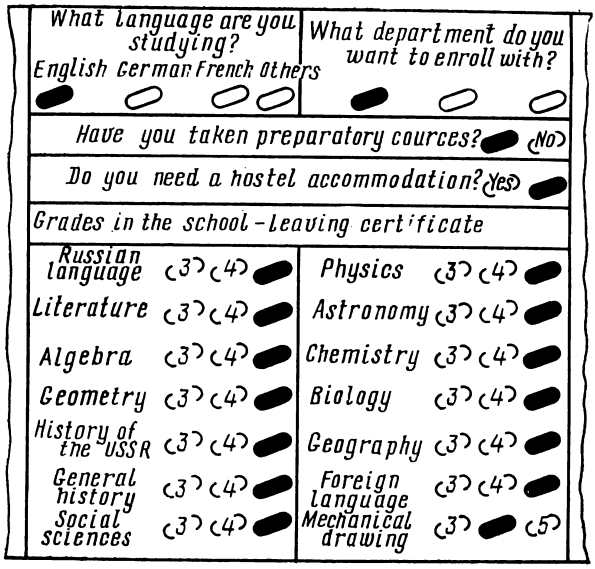
What language are you studying?
English German French Others
What department do you want to enroll with?
Have you taken preparatory courses? (No)
Do you need a hostel accommodation? (Yes)
Grades in the school-leaving certificate
Russian language (3) (4)
Literature (3) (4)
Algebra (3) (4)
Geometry (3) (4)
History of the USSR (3) (4)
General history (3) (4)
Social sciences (3) (4)
Physics (3) (4)
Astronomy (3) (4)
Chemistry (3) (4)
Biology (3) (4)
Geography (3) (4)
Foreign language (3) (4)
Mechanical drawing (3) (5)

Fig. 6.27. Part of a dual (mark-sensed) card

A deck of mark-sensed cards is loaded into the card hopper, *CH* (Fig. 6.28). When the read-punch is switched on, the picker knives, *1*, separate the lowermost card from the deck and advance it through the throat towards the first pair of continually rotating drive rolls, *2*, which grip the card and place it between intermittently rotating rolls, *5*. Moved by these rolls one step at a time, the card passes through a mark-sensing brush assembly, *6*, and enters the punching station, *PS*.

Each mark is sensed by an assembly of three brushes (there is a total of twenty-seven of such triple brushes). The two outer brushes are held at a positive potential of 110 V (Fig. 6.29). The middle brush is connected to the grid resistor of the associated thyratron. As the card passes through the mark-sensing station, the brushes sense the current-conducting marks, *M*, and complete

a voltage-divider circuit composed of the outer brushes at +110 V, the mark resistance (a few tens or hundreds of kilohms), and the middle brush at 40 V, with a resistance of 750 kilohms. As a

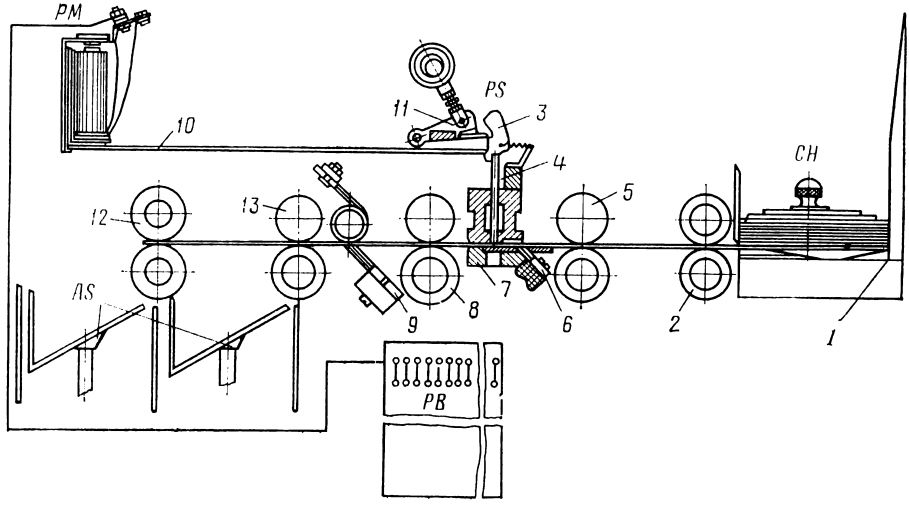

Fig. 6.28. Path of a dual card through a punch

result, a high drive potential is applied to the thyratron grid, the thyratron fires and draws a plate current which energizes the punch magnet, *PM*, placed in the plate circuit by switching contacts, *SwC*.

The punching station, *PS* (see Fig. 6.28) consists of a punch bail, *11*, a die, *7*, punch blades, *4*, and pawls, *3*, which are coupled by links *10* to the punch magnets, *PM*. When a particular punch magnet is energized, the associated link actuates its pawl so that its lobe is placed under the punch bail travelling to and fro. The bail strikes the pawl, and the associated punch blade punches a hole in the card.

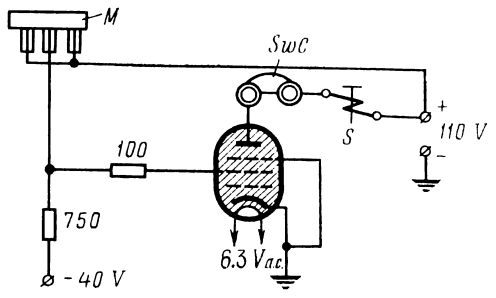

Fig. 6.29. Connection of reading brushes

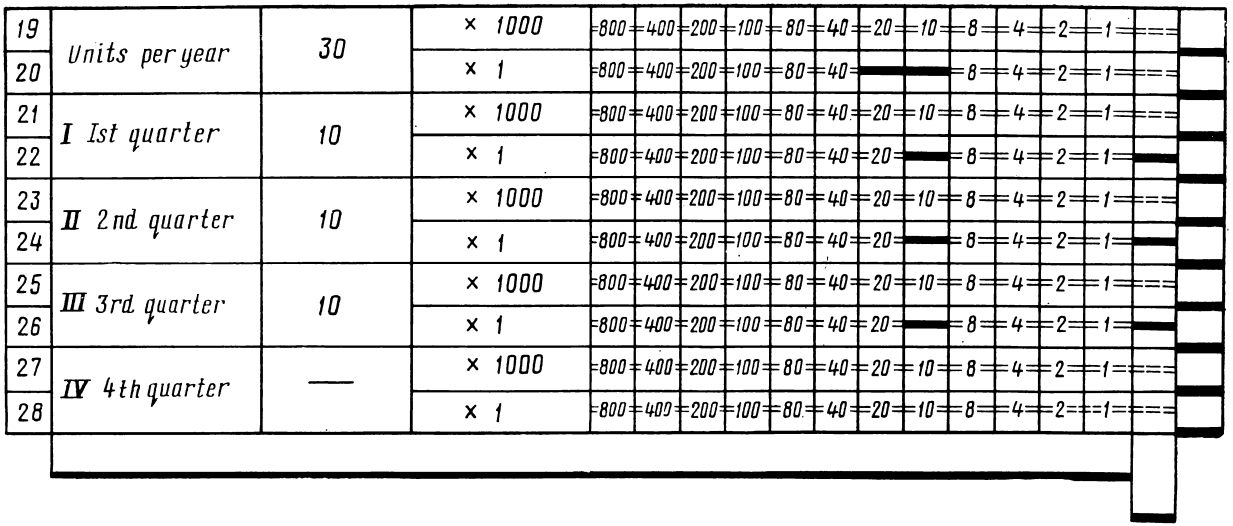

Fig. 6.30. Part of encoding form

The punching station has a total of 80 punch magnets (as many as there are punching positions in a card), each of which may be connected to any of the twenty-seven thyratrons via jacks on a patchboard. Thus, data entered in the form of twenty-seven marks may be punched into any twenty-seven columns of a 80-column card.

After the punching has been completed, the card is passed on to the reading station, *9*, of a sorter, located between two pairs of drive rolls, *8* and *13*. When the sorter magnet is energized, its lever is released and allowed to take up a position in which the card is directed to the first ("accept") stacker, *AS*. When the solenoid fails to be energized (which happens in the case of an unreadable information), the card passes by the first stacker and is carried to the second ("reject") stacker by another pair of drive rolls, *12*.

The ПС-80 can also monitor mark-sensing, duplicate from master cards, and punch totals. A particular mode of operation can be selected with appropriate switches on the control panel and using appropriate jacks on the patchboard, *PB*. The machine can handle 100 to 120 cards per minute, and the hopper can hold as many as 700 cards.

The Blank read-punch is intended to read data from an encoding form and enter them into a digital computer. An encoding form is printed on a white sheet of paper measuring 210 × 297 mm, which can accommodate up to 984 positions (24 positions across the width and up to 41 positions along the length). Each position can contain a numeral, a letter, or a mnemonic. The characters are drawn in soft or medium-soft lead pencil as bars at least 1 to 1.5 mm wide. Data fields may be given any convenient shape.

Figure 6.30 shows a part of a coding form stating the requirements in manpower for a year and each quarter of a year. The total manpower required is 30. So that this number can be read automatically, electrographic marks are made in the "20" and "10" columns. The quarterly requirement is 10, which fact is represented by a mark in the "10" column. Each line has two auxiliary positions, one for synchronization (the last column), and the other for an even parity check (the last but one column). The Blank equipment can handle 250 documents per minute, and its hopper can hold up to 700 forms.

6.7. TELEPRINTERS

A *teleprinter* * is an electromechanical device that transmits and receives messages or data over a two-wire line, using a certain signalling code. Advances in data processing and computer engineering have opened up a new field of application for teleprinters as a means of input preparation and entry into a computer. To this end, adjuncts have been added to punch and read data onto and from tape.

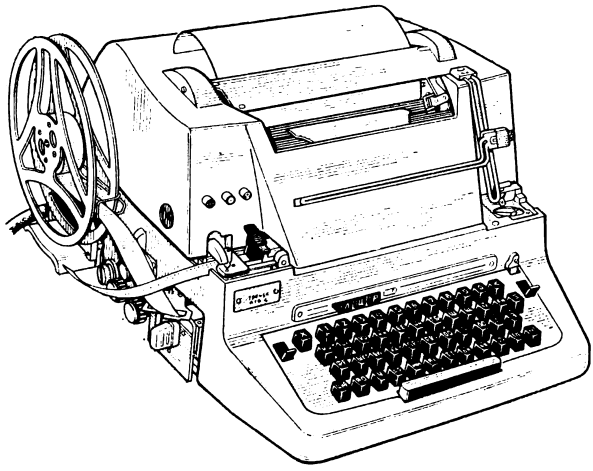

Fig. 6.31. Type PTA-6 roll teleprinter

A teleprinter may be either tape-printing or page (or roll) printing. A tape teleprinter prints a message or data on a narrow paper tape in a line fashion. A page teleprinter prints a message or data on a roll (or web) of normal-width paper, line after line just as an office typewriter does. Tape teleprinters have had a limited use in data-processing and computer systems, as tape is only convenient for short messages and becomes a nuisance where large blocks of data are involved. This is why page printers are predominant. Among the Soviet-made machines, the most commonly used one is the PTA-6 (Fig. 6.31).

PTA-6 teleprinter. Functionally, it has two independent sections, a sending keyboard and a receiving printer, set up on a common

* The word *Teletype* is a registered trade-mark owned by Teletype Corporation of Chicago, USA, applying to its manufactured line of printing telegraph equipment. — Tr.

frame, driven by a common motor, and controlled by a common electric circuit.

The sending keyboard converts each character to be transmitted into a five-unit code combination in accordance with the No. 2 International Code and sends it sequentially into line. The receiving printer receives and decodes the encoded signals, imprints the respective characters, and prepares punched tape for transmission.

KEYBOARD UNIT. This unit comprises a sending keyboard, a distributor, a tape distributor-transmitter, and an answerback feature.

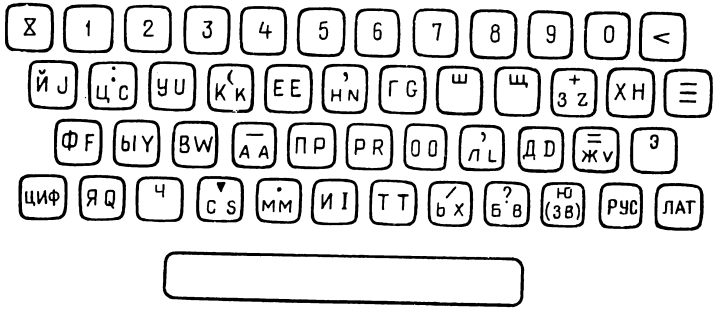

Fig. 6.32. Keyboard of the PTA-6 teleprinter

The *sending keyboard* consists of a keyboard proper, an assembly of intelligence selector (or pulse code) bars, a start bail, and a distributor start lever.

The keyboard (Fig. 6.32) looks like that of an office typewriter (the keys are arranged in four rows and attached to key levers).

The *distributor* in the PTA-6 teleprinter is a single-contact type which can operate on a neutral (single-current) or a polar (two-current) basis. It consists essentially of a start-stop mechanism, a distributing mechanism, and a contact mechanism. The start-stop mechanism starts and stops the distributor cam shaft in order to transmit a single character (code combination). The distributing mechanism converts the parallel presentation of each code group into a time-sequential form for transmission over a line. The contact mechanism generates the electric signals utilized by the distributor.

The *distributor-transmitter* uses a pre-punched tape for automatic transmission of messages or data. The prepared tape causes

the pulse code bars to set up on intermediate arms a mechanical combination representing the character to be transmitted. The necessary mechanical combination is set up as sensing pins "sense" the tape for code holes and complete the circuit to the associated contact when one is found. The tape is advanced to the next reading position by a tape transport mechanism. The distributor-transmitter also has a device to mark the used tape and a feature to stop the distributor-transmitter at the end of (or in the case of a break in) the tape.

The *answerback feature* automatically sends back an acknowledging combination in response to the "*Who are you*?" signal from the opposite end.

THE RECEIVING PRINTER UNIT. As already noted, this unit receives and decodes code groups, prints the respective characters on a web of paper, and punches them into paper tape. It consists of a receiving mechanism, a stunt box, a printer, a reperforator, and an automatic stop feature.

The *receiving mechanism* comprises a receiving magnet complete with an armature and adjustments, and a receiving selector mechanism complete with a start-stop distributor and a range (margin) finder. The receiving magnet has an U-shaped core and two coils. One arm of the armature lever interacts with the selector mechanism, and the other with the magnet. The magnet is matched to the line by adjusting the air gap between the armature and core.

The selector mechanism is a drum carrying on its circumference five pins and selector bars. The drum hub mounts a disc with five cams to force the armature towards the magnet during the reception of d. c. pulses making up a particular character. The selector is controlled by the start-stop distributor. The receiving mechanism is coupled to a transfer mechanism, the printer decoder, and the printer stunt box.

The transfer mechanism transfers the received code group from the selector to the decoder and stunt box of the printer.

The *printer decoder* positions the type wheel with respect to the printing hammer so that the character whose code is set up by the selector mechanism will be printed. A distinction of the PTA-6 teleprinter is that its printer operates in an inverse fashion. The principal element of the printer is a cylindrical type wheel on which the characters are arranged in three rows (which is an indication that the printer uses three cases). Shifting between the

cases is accomplished by moving the type wheel along the axis of rotation relative to the printing hammer.

The stunt box decodes special code combinations representing service functions associated with the printing action and some logic functions. These are figures/letters shift, line feed, carriage return, forward and backward spacing, bell actuation, identity signal release, and logical carriage return.

The sequence of operations performed by the receiving printer unit commences when a "start" pulse energizes the receiving magnet. The "Start" pulse is then followed by intelligence pulses. At the instants precisely timed with arrival of these pulses, the armature is forced by the respective cam towards the magnet core to be held attracted when the coil is energized or released when the coil is de-energized. The respective character is printed as the printing hammer strikes the ink ribbon against the paper.

The *reperforator* punches code holes in standard tape in accordance with incoming code groups. It consists of a cam shaft complete with a clutch and a throw-out mechanism, a selector mechanism, a punch station, and a tape feed.

The cams on the shaft actuate appropriate reperforator elements and the clutch which couples it to the start-stop distributor of the stunt box and transmits rotation from the drive.

The selector mechanism transfers the code set up on the bars of the stunt box onto the code bars of the reperforator. It has five code bars and sensing levers.

The type punch is a lever type; it punches holes in tape sequentially. The tape feed of the reperforator is similar to that used in the tape transmitter and steps the tape along at a constant pace.

All elements in the sending and receiving units of the PTA-6 teleprinter are actuated by an electric-power drive made up of an electric motor and gearing.

Of late, the Soviet switched public telegraph networks have widely been using the T-63 (RFT) page-printing start-stop teleprinters of GDR manufacture (Fig. 6.33). This three-case machine is a modification of the T-51 two-case teleprinter in turn adapted from the CTA-2M teleprinter. In addition to Russian characters, its keyboard has Roman letters. The characters are coded in International Alphabet No. 2. The power drive is made up of an a.c./d.c. commutator-type motor which actuates the shafts of the sender, receiving unit and printer. The speed of the motor is maintained by a relay regulator. The motor shaft mounts a worm, and

the driven shafts are fitted with fabric-base-laminate herringbone gears. The sending keyboard has 46 ordinary keys and one extended key, arranged in four rows. Under the key levers of the

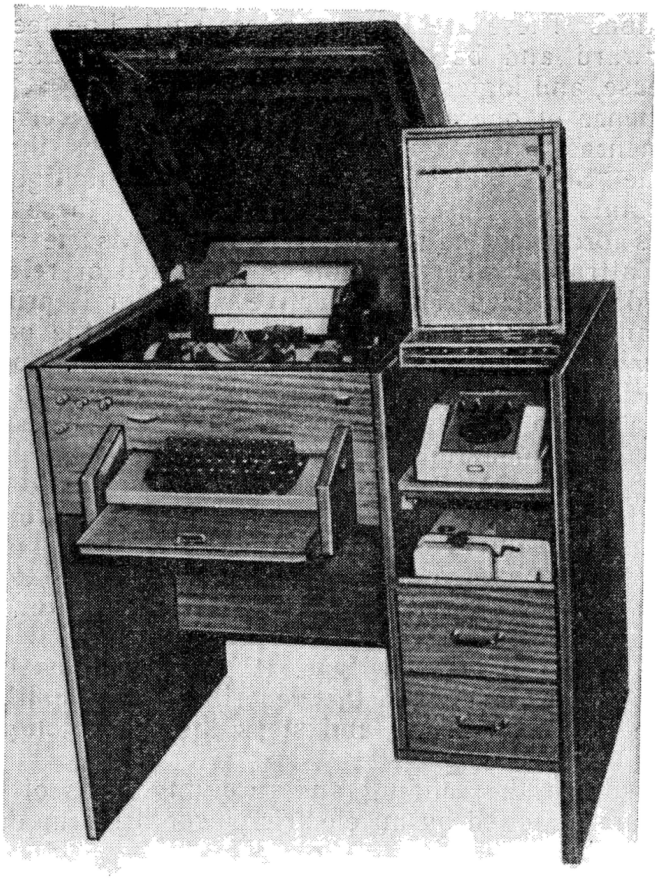

Fig. 6.33. Type T-63 teleprinter

keyboard are arranged the suppression (code) bar, the start bar, and the five intelligence selector (or code) bars, physically similar to their counterparts in the PTA-6 machine. The sending unit of the teleprinter consists of a contact system, a distributor, a start-stop mechanism, and a lock-out mechanism. The sending distributor has six pairs of contact springs actuated by the cams

on the distributor shaft. The distributor is started and stopped by the start-stop mechanism.

The answerback feature automatically releases the identity code set up on the code drum in response to the "Who are you?" signal from the calling station.

The teleprinter may be supplied with a T-63 tape transmitter adjunct (Fig. 6.34). It automatically transmits messages or data

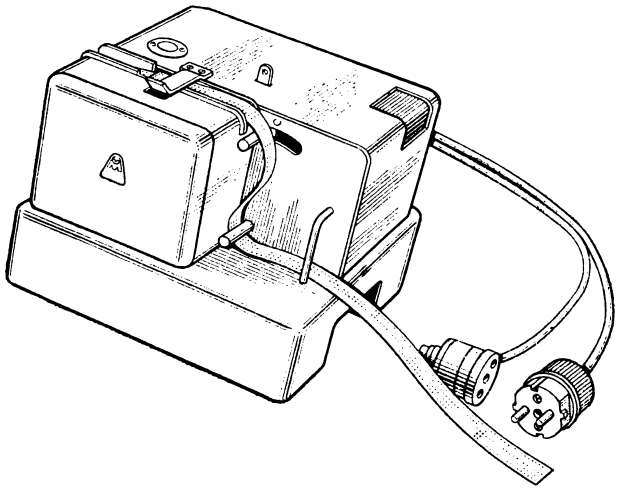

Fig. 6.34. Type T-63 tape transmitter

from a manually or automatically prepared tape. It has a selector mechanism, a distributor, a tape feed, and a power drive. The code combination to be transmitted is set up on the selector bars according to the hole pattern sensed by their pins in the punched tape. After the set-up code has been transmitted, the sensing bars are "knocked down" by suitable cams on the distributor shaft and by a knock-down bail. When a message has been completely read from the prepared tape, the tape tightens so that it might tear or break unless a device called the autocontrol lever stops transmission. When the teleprinter is permanently connected to another machine, it can be turned on and off remotely; when connection is via switching centre, this is done by a supervisory signal.

The receiving unit of the T-63 teleprinter incorporates a receiving magnet, a correcting start-stop mechanism, a selector me-

chanism, a printer, a range (margin) adjustment, and some ancillary elements. The receiving magnet has two coils and five armatures each carrying a contact spring. When a "Start" or a "Stop" pulse comes, the magnet actuates the start-stop mechanism. During the reception of intelligence pulses, it operates the selector mechanism. Each armature tends to break away from its magnet under the action of its spring. The five armatures are forced in turn towards the pole pieces of the magnet each time a code group representing a character is received. This is done by the lock lever of the clutch mounting the selector bars. With this arrangement, the magnet is only intended to hold the armatures already selected. This is why the selected armature is held at the respective pole piece upon arrival of a mark signal and is repelled by the spring in the case of a space signal. Arrival of a "Start" pulse allows the magnet armatures to drop out under the action of their springs and starts the receiving distributor; in the "Stop" condition the magnet holds attracted all the five armatures. The selector mechanism converts the received code group into a mechanical combination of decoding bars in space.

When the bar combination is complete, the printer prints the respective character. To this end, the printing cam is actuated to drive the respective type bar via a train of intervening elements. The type bar strikes the ink ribbon and images the character on paper.

The common functions such as line feed, carriage return, word space, automatic stop, etc. are performed in the T-63 machine in much the same way as in the PTA-6 teleprinter.

A tape perforator may be built as a self-contained adjunct for use in conjunction with any teleprinter or as a stand-alone unit. It has a selector mechanism, a punch station, a tape feed, a tape return and lock mechanism, a start-stop distributor, and a power drive. In operation, the punch station makes holes in the tape for the code combinations representing the characters keyed in on the keyboard of the machine. All mechanisms of a perforator operate along the same lines as their counterparts in the PTA-6 teleprinter.

6.8. DATA LOGGERS

At present, wide use is being made of equipment and facilities that serve to mechanize and automate the operations involved in data acquisition and reduction right at the place of origin (in production departments of a factory, within the various sections

of a single department, a warehouse and elsewhere), so that the data can then be directly fed to a computer for further processing. Such facilities are called *data loggers*. They may be units specifically designed for this function, or their functions may be performed by punched-card or punched-tape input preparation equipment.

As a rule, data loggers make use of standard "building blocks", such as a perforator, a keyboard, a punched-card or punched-tape reader, a printer, a data transmitter, etc. The actual set-up depends on the end use of the data logged. For example, the Soviet-made РП-10 data logger which puts out logged data on punched tape incorporates a punched-card reader, an identity card reader, a numerical keyboard, a numerical printer, a paper-tape punch, a control unit, and a selector unit. On top of these elements, the РП-50 digital data logger has a digital visual display and an adder to carry out the operations of addition and subtraction.

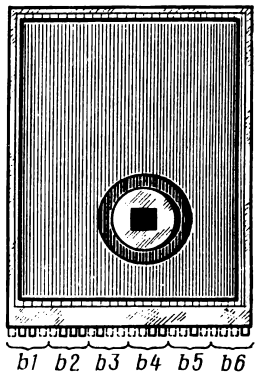

Fig. 6.35. Identity card

The РП-100 alphanumerical data logger can read in and out alphanumerical data. Physically, it is built into a two-pedestal desk. The data logged by the equipment may be classed into fixed, semifixed, and variable.

Fixed data include any standard rates, norms and similar information pre-punched on a card or cards, the identification No. of a worker indicated on his identity card.

The identity card is a rectangular piece of a thin but sturdy material on which the identification No. is recorded as a pattern of ridges and valleys (Fig. 6.35) in a binary-coded decimal (BCD) form. The code is read by the identity-card reader, ICR (Fig. 6.36).

Semifixed data may include the No. of a department, the No. of a section, the No. of a data logger, etc. These data are set in with the aid of ten-position microswitches on the selector panel of the semifixed data unit, *SFDU*.

Punched cards are read by a punched-card reader, *PCR*.

Variable data include those keyed in from the keyboard of a *Consul-254* electric typewriter.

A program to control the generation of service signals (such as "*Start of message*", "*End of message*", etc.), and the entry of some ancillary data ("*No. of message*", etc.) is recorded on a *program* card which is a normal 80-column type. This card is read by a program reader, *PR*.

The program reader consists of a sensing-brush assembly and a control unit. Initially, when no card is inserted in the reader,

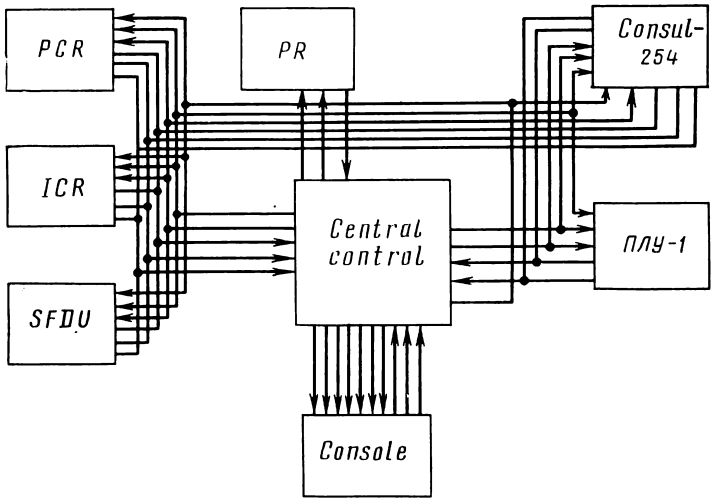

Fig. 6.36. Block diagram of an alphanumeric data logger

all contacts of the sensing-brush assembly are closed. When a card is inserted, the card feed is turned on to step it along under the sensing brushes. As the card is moved on a step at a time, the brushes sense the data column by column.

The above assortment of devices and the break-down of functions among them simplifies the entry of source data.

The fact that each worker bears an individual identity card and that any data he may enter in the logger will bear his No. enhances his responsibility for the correctness of the information he supplies.

The data logger incorporates a ПЛУ-1 tape punch complete with a control unit of its own. The tape punch converts all incoming data to a seven-unit code (to GOST 10859-64 or GOST 13052-67), with an eight digit position added for an even parity

check. Digital data may be recorded on tape, using five-unit international alphabet No. 2.

All input-output devices are controlled and interlinked by a *central control.*

The central control has four channels to accept data from input generating devices (the punched-card reader, the identity-card reader, the semifixed data unit, and the keyboard of the Consul-254 typewriter), and two channels to deliver data to output receiving devices (the Consul-254 typewriter and the ПЛУ-1 tape punch). Accordingly, the signals coming from input devices may be designated "S" (for "source") and those going to output devices, "D" (for "destination").

All signals involved in the transfer of data between input and output devices (commonly called peripherals) may be classed into two groups. One group covers transfer control signals, and the other group includes the transfer signals.

The principal signals in the first group are those used to alert a particular peripheral or to mark the end of transfer. Those in the second group are "*Transfer data*", "*Transfer instructions*" and "*Transfer status*".

The control signals, "*Call to input device*" (C → S) and "*Call to output device*" (C → D) are always directed from the central control to the input or output device involved. The control signals, "*End of transfer from input device*" (E → S) and "*End of transfer from output device*" (E → D) are always directed from the peripheral device involved to the central control. In other words, the control sequence involved in the transfer of data between the peripherals and the central control may be defined as C → S, C → D, E → S, and E → D.

Consider the sequences of control signals involved in an operating cycle.

Control state I:

$$C \rightarrow S = 0, \qquad C \rightarrow D = 0$$
$$E \rightarrow S = 0, \qquad E \rightarrow D = 0$$

(where the zeroes indicate the absence of any signals).

All input and output devices assume this state when the logger is just turned on ("*Initial setting*") or after the previous sequence has been completed. In this state, the central control is brought to readiness to sample its peripherals and sends out the signals C → S and C → D to the input and output devices standing next on the program.

Control state II:

$$C \rightarrow S = 1, \qquad C \rightarrow D = 1$$
$$E \rightarrow S = 0, \qquad E \rightarrow D = 0$$

(where the 1's indicate the presence of signals, and the 0's, their absence).

In this state, the selected input device transfers its data to the central control, and data from the central control are transferred to the selected output device. If the data reach their destinations, the output device generates the signal $E \rightarrow D$, and the input device, the control signal $E \rightarrow S$.

These signals are generated at the instant when a given state terminates and the next commences.

Control state III:

$$C \rightarrow S = 1, \qquad C \rightarrow D = 1$$
$$E \rightarrow S = 1, \qquad E \rightarrow D = 1$$

In this state, the data generated by an input device is processed by the central control. At the instant when these data are no longer needed, the central control cancels the signal $C \rightarrow S$. On the other hand, this state indicates that the data available at the output of the central control are no longer accepted by the output device and may therefore be modified or cancelled, in which case the central control cancels the signal $C \rightarrow D$.

Control state IV:

$$C \rightarrow S = 0, \qquad C \rightarrow D = 0$$
$$E \rightarrow S = 1, \qquad E \rightarrow D = 1$$

In this state, neither the input nor the output devices are in a position to receive new signals, $C \rightarrow S$ and $C \rightarrow D$, from the central control. The output device is completing work on the previous batch of data, and the input device is getting ready to supply new data. At the instant when the output device is ready to receive data, and the input device to supply them, the signals $E \rightarrow D$ and $E \rightarrow S$ are cancelled. This brings the control sequence in the channels of the input device, S, and output device, D, to an end.

The data logger may be operated manually or automatically, by placing the "mode of operation" wafer switch in an appropriate position. In manual operation, the program punched card is not utilized, and a particular input device is selected by pres-

sing the respective control button, and a particular output device, by throwing its toggle switch to the "on" position.

In automatic operation, all devices are scanned (or sampled) in accordance with the instructions keyed into the program punched card.

These instructions (such as "*Start of message*", "*End of message*", "*Status test*", etc.) are read from the punched card into the central control automatically. After a message has been pro-

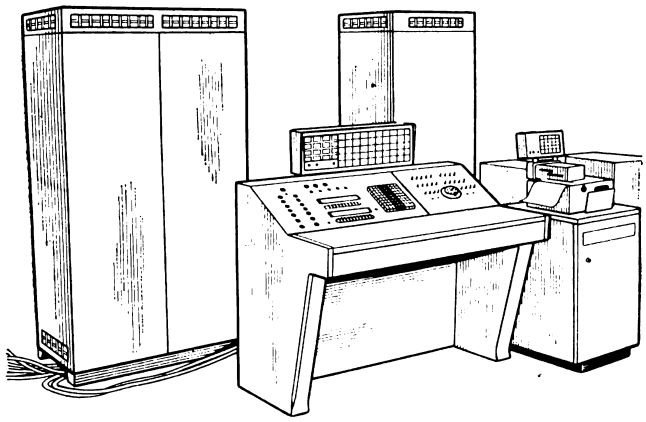

Fig. 6.37. Type АРП-1М data logger

cessed, a "*Clear*" instruction is fed to all devices, the devices are reset, and the data logger is in readiness to process the next message.

At the end of a shift or at any other time, the punched tape is transferred to a computer centre to be processed automatically by a digital computer.

An example of an automatic data logger of Soviet manufacture is the АПР-1М (Fig. 6.37). It can acquire, process and record data from as many as fifty locations at a mass-production factory, including down-time of plant and product output, prepare machine documents holding shift totals, summon the personnel involved, and has telephone circuits.

The complete data logger includes fifty remote consoles, the control engineer's console, a central control incorporating an internal memory, a ПЛ-80 tape punch, an АПМ-2М electric typewriter, as many as eight light annunciators, and a power unit.

Each remote console has eight three-way toggle switches to turn on a call signal and down-time indication, and a toggle switch to indicate the end of repair. The end of down-time or of repair is indicated by throwing the respective toggle switch to the OFF position.

Physically, the control engineer's console is a combination of a desk, a vertical panel, a dial, and annunciators with which he is in a position to select a particular mode of operation for the data logging system, order data readout, and monitor the system's operation.

The central control unit is a cabinet housing internal memory and the circuits necessary to control data acquisition, storage and recording. The memory has a capacity sufficient to store data likely to accumulate over a shift (usually a hundred 13-digit decimal numbers). Each remote console is allocated two such numbers, that is, 26 decimal digit positions.

The tape punch prepares a tape presenting totals over a shift (the data are keyed in BCD form).

At any time, the control engineer can actuate his console typewriter to print out data in tabular form, give product output and down-time due to all likely causes for each remote console.

CHAPTER 7

DATA TRANSFER SYSTEMS AND DEVICES

In an automatic management information system, data are collected from a great number of sources and transferred to a computer or a computer centre for further processing. Data acquisition centres may be equipped with a variety of devices to convert data to a form convenient for entry into a computer. Inevitably, the data sources and the data converters will differ in speed of operation, amount of information generated or handled, and data priority. This is why it is important to set up input/output devices and the associated computer in a system where data would be transferred in the best possible manner.

7.1. DATA TRANSFER IN FIRST-GENERATION COMPUTERS

In a very simple case, the data transfer system would incorporate a punched-card and/or punched-tape reader, *CR* (*TR*), and a printer (Fig. 7.1). This type of data transfer system was used with first-generation computers.

In this system, data were handled as follows. Data read from a punched card or tape were put into a transfer control unit, *TC*, which routed them to a processor in response to appropriate instructions. Computed results were made available to the user via the same transfer control unit and the printer, *CP* (*TP*).

The transfer control unit also served external magnetic storage, *M* (magnetic tapes or drums).

It was characteristic of first-generation computers that no operations were carried out in the machine as long as data transfer was under way; that is, the central processor was standing idle.

The above system may be called an internal transfer system, because the manufacturer would supply a computer complete with *I/O* devices, magnetic storage and a printer, and these would be operated as part of that particular computer. Data could only be exchanged within its limits and had no external inlets or out-

lets, except in document form. In fact, every source document had first to be keyed into punched cards or tape, and these would then be sequenced by a human operator.

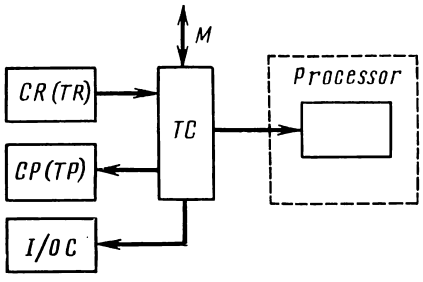

Fig. 7.1. Data transfer in a first-generation computer

The speed at which source data could be entered in the computer was limited by the capability of input devices. An input cycle would then be followed by a computation cycle, and the latter, by a readout cycle. The total time was the sum of the terms which were functions of the operating speeds of input devices, transmission links, computer, and output devices. Obviously, this arrangement could not but entail a sizeable loss of machine time during data exchange.

7.2. DATA TRANSFER IN SECOND-GENERATION COMPUTERS

An internal data transfer system is convenient for a computer centre, but inadequate for a management information system where data come in from a variety of sources. In a management information system, the great number of peripherals should be combined into an integrated exchange system. The latter may be an internal transfer system extended to include an external transfer system (Fig. 7.2) or it may be an independent, ramified exchange system (Fig. 7.3) in its own right.

In a ramified transfer system, one may discern channels and subchannels.

A *transfer channel* is a collection of facilities which control data transfer between the peripherals and the associated processor. A *transfer subchannel* is a collection of facilities which maintain data transfer between a particular peripheral and the transfer control.

In a data transfer system using a great number of peripherals, it is important to take appropriate measures so as to ensure high efficiency in the utilization of data and the computer.

One approach is to carry out data transfer and data processing concurrently. This is achieved through program interrupts — the execution of the main program is interrupted at suitable instants to execute another program (or programs).

Another approach is to interleave several *I/O* operations so that several subchannels (that is, *I/O* devices) can be served by the processor during the same overall time. In this case, the total number of subchannels (or *I/O* peripherals) that can be served during the same overall time is limited by the speed of the computer.

This approach has led to grouping *I/O* devices (subchannels) according to data rates. Low-speed *I/O* devices operating simul-

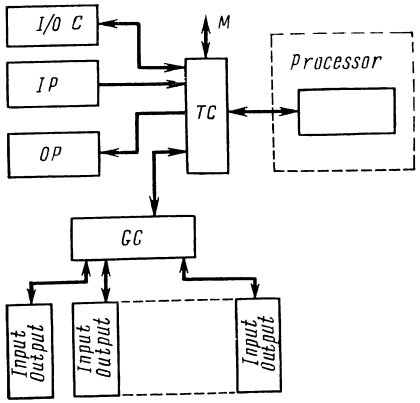

Fig. 7.2. Internal and external data transfer systems

IP—input card punch; *OP*—output card punch; *TC*—data transfer control; *M*—memory; *I/O*—input-output devices; *GC*—group commutator

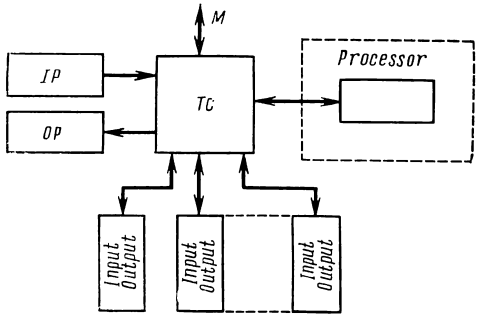

Fig. 7.3. Ramified data transfer system

IP—input card punch; *OP*—output card punch; *TC*—data transfer control; *M*—memory

taneously and asynchronously of one another are controlled by the *multiplexor channel.*

High-speed *I/O* devices which cannot be served by the processor all at the same time are under control of the *selector channel.*

Consider the data transfer system of second-generation computers in greater detail.

So that the high speed of the central processor can be utilized to advantage, it is usually supplied with a wide range of external storage and *I/O* devices. As already noted, these devices are under control of multiplexor and selector channels. As a rule, magnetic-tape, magnetic-drum and magnetic-disc storage devices are controlled by selector channels, while punched-tape and punched-card readers, tape and card punches, and teleprinters by multiplexor channels.

All channels are served and linked to the central processor, *CP*, by a *I/O* or transfer control unit. In Soviet-made second-generation computers (such as the БЭСМ-6, the Ural-14 and the Minsk-32), the link between the peripherals and the central processor is provided by a control unit consisting of a peripheral control unit, *PCU*, and peripheral selectors/distributors, *PSD* (Fig. 7.4).

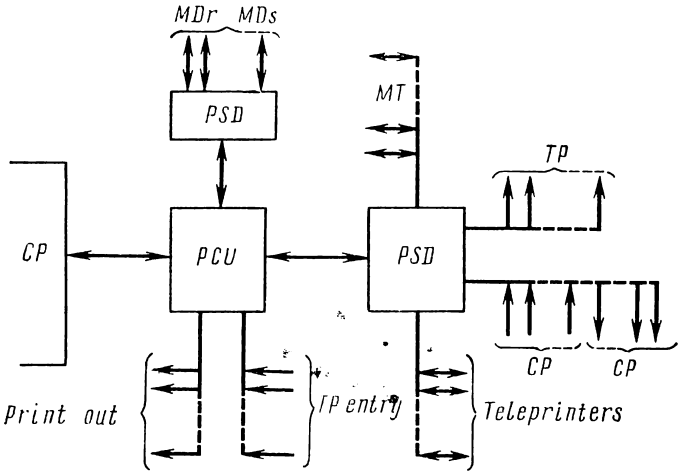

Fig. 7.4. Data transfer in a second-generation computer

The peripheral control unit holds all the basic circuits over which the peripherals are linked to the central processor and referenced in response to appropriate commands. This unit also provides a link with the peripheral selectors/distributors, *PSD*, the printers, and the high-speed tape readers. In turn, the peripheral selector/distributor rack links the peripheral control unit to all magnetic-tape, *MT*, magnetic-drum, *MDr*, and magnetic-disc, *MDs*, storage units, the card punches, *CP*, the tape punches, *TP*, and the teleprinters.

Basically, the peripheral control unit is a random-access storage in which each location is a buffer register assigned to a particular peripheral device. The information written in a buffer register may be a control word (if an action is initiated by the peripheral device) or output data for the associated peripheral device. Data are written into and read out of the buffer registers in response

to instructions by which the respective peripheras are referenced.

A reference instruction comes to the peripheral control unit from the central processor. The address adder generates the execute address which goes to the instruction address unit along with the instruction code and write or read code. From the output register of the instruction address unit, the execute address of the reference instruction, which is the address of the peripheral device involved, is fed to the peripheral address register in the peripheral control unit. One of the stages in this register stores the instruction type code; the remaining stages are utilized as decoders to generate the adress (number) of the peripheral device and the associated buffer register.

Transfer of data between a peripheral device and the processor is commenced after the computer sends a signal to the peripheral control unit and the latter acknowledges its receipt. In the case of a read instruction, the computer receives the code from the peripheral control unit (that is, the contents of the location whose address has been specified in the reference instruction). In the case of a write instruction, data are transferred from the computer into the control-unit location whose address has been selected by the decoders.

7.3. DATA TRANSFER IN THIRD-GENERATION COMPUTERS

Advances in computer engineering and data communication systems have led to the appearance of multiple computer systems.

Multiple computer systems are a logical outcome of two general trends in the computer field — the desire to build computers on a modular basis and the desire to run all elements of a computer at the same time. These two trends have been responsible for the fact that the various modules of an individual computer have grown into computing facilities in their own right, capable of handling various aspects of a major problem or the various problems of a single system all at the same time.

The heart of a multiple computer system is the host computer which has under its control subordinate, although self-contained, satellite computers. Both the host computer and the satellite machines can, and do, use a variety of *I/O* devices of their own. This is the reason why third-generation computers are often called *computer systems.* It is to be noted, however, that there may be computer systems organized on a different basis than described above.

The *I/O* devices controlled by selector and multiplexor channels have become still more varied than they were before. Such a multitude of peripherals are now linked through an *interface.* An interface is a collection of hardware and software which provides the necessary matching in order to make successful the operation of the interconnected functional entities of an automatic data processing system.

The third-generation satellite computers have each an internal storage of its own and execute each what are called subsidiary programs (written into its internal memory). These routines replace some of the hardware required for operation of the satellite computers. In multiple computer systems, when a particular satellite unit is through with its job an appropriate flag is written into what may be called a *duty* register. The central control unit scans the duty register every clock period, that is, each time the central processor completes its cycle, and takes notice of its contents.

It follows from the foregoing that a third-generation computer is in effect a system of several machines each of which has a control unit, an instruction set, and an operating cycle of its own. As a rule, the individual satellite units (which may be called data processing terminals, or DPT, for short) are spaced great distances apart in a data processing system. This is why data communication links have to be used in order to transfer data from a d.p. terminal to the central processor or to another terminal. At present, the best way to match a computer to communication channels is to use computer systems each of which incorporates one or several high-capacity data-processing computers (or *central processors*) and special-purpose auxiliary communication computers (or *communication processors*).

Since they are special-purpose machines, communication processors usually have a limited capacity and a narrower range of logic capabilities as regards data processing than central processors. Any communication processor having an independent arithmetic unit, control unit and storage, can execute a special data-transfer program independently of the central processor; it is directly connected to a digital bus or highway over which the central processor communicates with all the other units of the computer system at a very high rate. The instruction set of a communication processor is ordinarily limited to anywhere between 10 and 20 special data-transfer instructions containing control characters of the microprogramming type, so that the in-

struction set may greatly be extended, should the need to do so arise.

An example of a computer specifically designed for data communication is the IBM-7741 developed by the International Business Machines Corporation of the United States. This is a high-speed program-controlled digital machine adapted specifically to handle a veriety of communications problems. Its capabilities may be summarized as follows:

it converts signals transmitted over communication channels into those needed for operation of the IBM-7741 and back;

it regenerates corrupted signals;

it converts the transmitted code into that of the central processor;

it provides a buffer between any communication links varying in handling capacity;

it tests communication lines and isolates the faulty ones;

it tests data-communication equipment;

it converts a serial to a parallel code and back (assembles characters from digits and breaks up characters into digits);

it assembles characters into messages and breaks up messages into characters;

it receives and transmits variable-length messages;

it establishes priority for message processing;

it switches complete messages between lines;

it analyses messages to a predetermined program;

it edits and produces messages (it removes service information from incoming messages and inserts it into outgoing ones);

it keeps a record of load on communication links;

it checks and corrects the data transmitted;

it follows priority rules;

it indicates time of day.

In performing all of these functions, the IBM-7741 widely uses both hardware- and software-controlled techniques. The machine cycle of the processor is made up of four memory cycles. The first memory cycle is utilized solely to scan the channels, and the remaining three, to execute the program and other functions involved in data processing.

A major machine cycle consists of 124 such machine cycles. It consists in the sequential scanning of 124 communication channels and is executed in 4960 μs. As digits come over a line, they are assembled into characters in appropriate memory locations assigned to each communication channel. These are channel lo-

cations. From them, the characters are transferred into the memory files referenced by the addresses stored in other locations associated with the channels. These are *second channel locations.*

As soon as a message is written in the memory of the IBM-7741 it is subjected to all kinds of processing provided for in the program, and transferred into the digital bus or highway.

To sum up, a distinction of the data transfer system used with third-generation computers is that its *I/O* peripherals are self-contained entities having their own storage, arithmetic units, control units and *I/O* devices. They can execute their functions to their own programs.

7.4. GENERAL PRINCIPLES OF PROGRAM INTERRUPTS

By definition, a *program interrupt* is a temporary break in the instruction currently being executed with provision for returning at a later time to the main program at the point where interrupted. A program interrupt system enables the central processor to respond with sufficient flexibility to external or internal events that may occur asynchronously at any instant. Each event is represented by signals $x_1, x_2, \ldots, x_n$ which are generated by the various elements of the digital computer and appear at the input to the program interrupt system as interrupt requests.

During operation, the computer uses appropriate hardware or software to keep continuous watch on these signals written in registers and pick them according to their priorities.

The program interrupt system:

enables the processor time to be utilized better;

simplifies real-time programming, as it allows greater independence in the preparation of various programs;

offers a ready means for breaking up a complex program into several independent segments and allows, at the same time, other programs to be executed during the same overall time;

permits the digital computer to select rapidly one of several service programs to handle malfunctions or errors in the main program;

enhances computer reliability.

In developing a program interrupt system for a particular digital computer, it is necessary to select a method by which the causes of interrupts can readily be identified; to time the interrupts; to define the sequence in which operations are to be performed in the case of an interrupt; to decide on the type of prio-

rity assignment; to organize individual control of interrupts; and to evaluate the performance of the program interrupt system.

Let us consider basic approaches to handling the above tasks with regard to present-day time-sharing data-processing systems.

Recognizing sources of interrupts. In a multiprogramming system, each interrupt source always has its own 'flag' that is set when it is requesting a program interrupt. The flag status is then analyzed by hardware or software.

In analysis by software, each interrupt flag is assigned a location in the computer memory. Initiation of a request for a program interrupt causes a 1 to be written into that location; otherwise it will contain a 0. The whole set of flag locations forms a tableau which is sampled by software to establish its identity. Once the interrupt source has been identified, a jump to the interrupt routine takes place. A disadvantage of the software-controlled recognition of interrupt sources is that it takes up too much time.

With direct hardware-controlled recognition, each interrupt source is assigned an interrupt subroutine of its own, and a jump to this subroutine is completely hardware-controlled. This approach increases the apparatus inventory of the system and commands a considerable proportion of the internal memory, but it is fast and convenient to the programmer.

Most often, resort is made to a combination of the two methods, each being given a specific part of the functions handled by the program interrupt system.

Timing of interrupts. Program interrupts may be timed in one of several ways as follows: (*a*) each source may be allowed to initiate an interrupt at any time; (*b*) each instruction may be arranged to contain a control character to enable or disable an interrupt upon its execution; (*c*) each interrupt flip-flop may be ganged up with its own protection flip-flop, and the whole set of protection flip-flops may form a protection register as a means of flexible control over interrupts on an individual basis; (*d*) interrupts may be handled as in the previous case, but they may also be disabled at particular instants of time.

Present-day computers are mostly using the methods defined in (*c*) and (*d*).

Chain of operations in the case of an interrupt. Whatever the manner in which interrupt sources are recognized, there are some basic functions that the program interrupt system will perform as it jumps to the interrupt routine. In many computers, the pro-

gram interrupt system operates as follows: it times interrupts; it disables any further interrupts while handling that on hand; it saves all information about the interrupted program for return to it at the point where interrupted; it generates the address of the first instruction in the interrupt-recognizing subroutine; it recognizes the interrupt source and calls an appropriate subroutine for execution; it restores the previous data (partial results) and returns to the interrupted (main) program.

Priority types. Multiprogramming digital computers use programmable (software-controlled), wired-in (hardware-controlled) and mixed priorities.

1. *Software-controlled priority.* This type is used in systems where interrupts are grouped into classes. Each interrupt source is assigned one bit in the interrupt-request storage register, and each interrupt class has its own bit in the interrupt register. Also, each interrupt class has its own bit in the mask (protect) register. The value of the mask register can be varied by a special routine. This arrangement provides flexible control of priorities.

2. *Hardware-controlled priority.* As in the previous case, interrupts are grouped into classes widely varying in number. Each class is assigned a specific priority level or rank. It is assumed that the first level commands top priority, and the remaining priorities decrease in importance with increasing numbers $(2, 3, \ldots, m)$. Obviously, any ith priority rank may interrupt any of the subroutines at the $(i+1)$st, $(i+2)$nd, etc. level, but it cannot interrupt a subroutine whose level bears a lower number. As a rule, a subroutine in the ith level cannot interrupt any other subroutine in the same level. This type of priority system is implemented by suitable hardware.

3. *Mixed priority.* In case (2) the level priorities are fixed by hardware, which fact subtracts from the flexibility of the system. This disadvantage can be avoided through a combination of cases (1) and (2), so that the priority of some levels can be changed by suitable commands which set appropriate protection codes into the mask register.

Evaluation of performance of the program interrupt system. The performance of a program interrupt system may be evaluated in terms of the following parameters:

the *response time*, t_r, defined as the time interval between the instant when a request for an interrupt arrives and the execution of the first useful instruction in the requesting program is begun;

the *idle time*, t_i, defined as the difference between the overall time required to process the incoming request, t_0, and the time expended to execute all useful instructions, t_c, $t_i = t_0 - t_c$;

optimum priority response, defined as the system's ability to handle a task which commands top priority at the moment. In other words, at any instant the computer must handle a task associated with a request having top priority:

$$\varphi_k = \max(\varphi_1, \varphi_2, \ldots, \varphi_n) \qquad k = 1, 2, \ldots, n$$

where φ_k is the weight factor representing the priority of the kth task;

system saturation, defined as the system's inability to respond in time to some requests, which leads to loss of some information;

the *cost of the program interrupt system.*

Let us have a closer look at how the interrupt system operates.

An x_1 interrupt may be caused by any computer malfunction, any corruption in an information word, violation of memory protection, selection of an invalid instruction code, selection of a decimal operand with an alphabetic combination, an illegal exit to a subroutine, etc. Obviously, these causes may arise from both improper hardware operation and improper programming. The control programs handling these interrupt requests analyse the situation and make an appropriate decision.

An x_2 interrupt is initiated when a hardware or software malfunction occurs in any one *I/O* channel. These malfunctions may be a corruption in an information or control word, a corruption of an information character, a failure in a specific peripheral device, a malfunction in a device during operation, etc.

An x_3 interrupt is typical of interactive peripherals engaged in an "off-hand" dialog with the computer. This signal is an indication that a device is to enter data into the computer. Control programs recognize the calling peripheral device by its number, clear space in memory for the incoming data, and signal the device that it is proper to enter its data.

User peripheral programs employ peripheral extracodes. Just as an extracode turns up in a user program, a special signal, x_n, initiates a jump to a service routine. This routine recognizes the interrupt source and transfers control to the supervisor (see Sec. 7.5).

7.5. LOGIC STRUCTURE OF TIME-SHARING SYSTEMS

The salient points of a time-sharing system may be stated as follows:

it makes its resources available to a great number of users;

each user is served in a manner entirely independent of how any other users are served;

the cost of services is kept to a minimum.

A time-sharing system may operate in any one of several modes, namely the remote batch-processing mode, the interactive processing mode, the conversational processing mode, and the universal mode.

In the *remote batch-processing mode*, the central computer processes data accumulated over a period of time from remote locations. The processed response is outputted after a time lapse (which may be hours, days or even weeks).

In the *interactive processing mode*, the machine is busy doing what is known as the background task which may be, say, batch processing. There is a number of remote terminals through which authorized users may send in a series of questions for each of which there is a program to generate an appropriate answer. When an inquiry comes from a remote terminal, the control program interrupts the background task and transfers control to the respective answer generating routine. The latter generates an answer which is immediately relayed to the originating remote terminal, and the machine resumes the background task.

In the *conversational processing mode*, the system's resources are shared between a limited number of users having direct access to the computer via interactive terminals (which may incorporate typewriters, CRT displays, graphic plotters, tape readers and teleprinters).

Systems operating in the *universal mode* combine all the modes defined above. Users have direct access to the computer through terminals which may be remote and linked with direct-access facilities (magnetic discs, drums, tape, etc.) located near the computer. The direct access devices are controlled by the central operator in response to requests from the terminals.

Multiprogramming has brought with it the need for sophisticated control programs. Used in conjunction with back-up hardware, these programs enable the system to handle tasks efficiently. Consider control programs as an hierarchy of two levels. The

top level is occupied by a scheduler and the bottom level, by a supervisor.

A *scheduler* is a planning program which schedules an optimum handling of several tasks on the basis of data supplied by the user and according to their priority ranking, urgency, expected cycle time, and the load on the peripherals.

A *supervisor* coordinates the execution of a stream of tasks and allocates to them the system resources (processor time, storage, *I/O* devices) on the basis of interrupt signals and the data supplied by the scheduler. The supervisor incorporates an *I/O* control program. Let us take up this program in greater detail.

The *I/O* supervisor performs input/output functions the most important among which are allocation of channel resources and *I/O* devices, initiation and termination of channel programs, software-controlled analysis of the channels and *I/O* devices, and correction of *I/O* errors.

The *I/O* supervisor is controlled via the program interrupt system. In order to link his program to the control program, the programmer uses system macroinstructions. As a rule, these are *I/O* and telecommunication macroinstructions.

When a request comes for *I/O* operations, the *I/O* supervisor tests to see if the requested devices and channels are free. If they are not, the request is put on the queue; if they are free, the channel program is started at once.

In its operation, the supervisor uses several control blocks, the principal among which are the input/output control block, the event-synchronization block, and the device control block.

The device control block generates control signals for each of the *I/O* devices linked to the system. Its information fields contain permanent information about each device (class, type, channel address, subchannels, device status, addresses of programs specific to a given device, error-correcting programs, queue and channel control, etc.).

The *I/O* control and event-synchronization blocks handle each request for *I/O* operations.

The *I/O* control block contains information essential for the execution of a channel program (its address, storage area holding information about the channel status, address of the user program, etc.).

The event-synchronization block times the progress in the execution of a program, the termination of *I/O* operations, and the very fact that data have been processed.

7.6. MULTIPLEXOR AND SELECTOR CHANNELS

Multiplexor channel. The purpose of a multiplexor channel is to provide a link between the processor and peripherals which operate at relatively low data rates (the peripherals are connected to the multiplexor channel via an interface). As regards performance, the multiplexor channel should be a good match to the computer. The number of peripherals connected to the multiplexor channel depends on the speed of the devices and the computer. In the Soviet-made EC-1020 computer, the multiplexor channel can attach up to 242 peripheral devices.

Actual data transfer (*I/O* operations) is accomplished by subchannels. These occupy an area in main storage, called the *multiplexor channel storage*. It stores and modifies all information essential to *I/O* transfers.

Functionally, the multiplexor channel of the EC-1020 computer consists of several registers which use interface lines and buses to control users, input and output data, and to monitor overall performance.

The multiplexor channel is coupled to the processor via a channel control unit which consists of channel control registers, a constant decoder, a control register, a microprogramming halt request generating circuit, and units which deliver the contents of the channel registers to the computer input and the contents of the channel data registers to the inputs of the information registers in the internal memory.

The function of a channel control register is to strobe the interface lines. It consists of a chain of flip-flops each performing a specific function. The flip-flops are set to a particular state by the multiprogramming or logic circuits of the channel.

A user control register is intended to store the input interface signals (user's operation, user's address, user's control, etc.), and also to generate an "interface free" flag, device-select signals, and a request for a microprogramming halt. It consists of flip-flops which are set and reset by hardware in response to appropriate signals.

The output data register feeds data to the channel buses. It has seven information digits and one check digit. Data can be written into the register from the information registers of the internal memory and from the arithmetic-logic unit. Each flip-flop is set by a signal coming from the respective digit position of the internal-memory register.

The input data register is the name given to the receiver amplifiers through which data from the user wires are transferred to the information registers of the internal memory and to the input of the logic unit. From the user wires, data go to the input of the logic unit only if the "operate" flag of the multiplexor channel is set.

One of the check registers is intended to examine interface flags and signals for validity and correspondence. It uses seven flip-flops which are set by signals over appropriate wires.

The other check register uses eight flip-flops each of which is set when an interrupt is recognized in operation of the interface.

In all channels, three forms of control information are used to perform input-output operations, namely instructions, commands and orders.

The *I/O* instructions are part of the program. All operations of the *I/O* devices in a channel are specified by four instructions as follows:

— "*Start input-output*", used to start all operations involved in data transfer and control;

— "*Test input-output*", used to determine the status of the channel, subchannel and addressed peripheral device;

— "*Stop input-output*", used to stop input-output operations;

— "*Test channel*", used to determine the status of the channel.

All instructions specify the channel No. and the address of the device involved.

The commands, of which there are six, are intended to initiate input-output operations on the peripheral device involved. Three of them, ("*Write*", "*Read*", "*Read back*") are set aside for use in exchange with peripherals.

Orders are stated in control commands and define the functions specific to a particular peripheral device, such as "*Rewind tape*", "*Set record density*", etc.

With a multiplexor channel, the input-output operations are initiated by the "*Start input-output*" instruction. The microprogram links the channel to the peripheral device involved and feeds the command code there. What happens next depends on which mode of operation has been chosen. This may be the burst mode or the multiplex mode. In the burst mode, a single *I/O* device captures the channel and does not relinquish it until the data interchange cycle is complete. In the multiplex mode, the single data path of the channel is shared by several peripheral devices

operating simultaneously. The channel receives and sends data from and to them, using microprogramming interrupts.

After a data interchange cycle is complete, the peripheral device generates a status byte, "*Channel over*" or "*Device over*".

Selector channel. The selector channel provides a link between the processor and high-rate peripheral devices, such as magnetic-tape, drum or disc storage units. These devices have data rates comparable with the cycle time of the internal memory and are connected to the selector channel via an *I/O* interface control unit. The number of channels, *I/O* interface control units and peripherals involved depends on the type of computer used. In the Soviet-made БЭСМ-6, the selector channel can attach as many as 16 magnetic drums and 32 magnetic-tape drives, while in the Minsk-32 the total number of high-rate devices is not over 32.

Since the peripheral devices controlled by the selector channel operate at data rates comparable with the cycle time of the internal memory, only one device can operate at each particular instant, but the search for the desired block or zone may be carried out on several magnetic media at a time.

When memory-reference requests come from several *I/O* devices at a time, they are served in accordance with the established priority. This, however, entails several delays in data interchange. The duration and nature of these delays depend on the manner in which data transfer is organized and the speed of the device(s) involved. For example, in the БЭСМ-6, four kinds of delay may occur:

(1) delay t_1 which occurs when an address code is transferred from the external-access buffer register to the magnetic-storage address output register where at least 0.3 μs is necessary for codes to be changed;

(2) delay t_2 which occurs when an external reference request comes later than some other request occupying the storage block to which external reference is to be made. Its maximum duration is 2 μs (which is comparable with the magnetic-memory cycle time);

(3) delay t_3 which occurs because it takes at least 0.6 μs for a code on the external-access address buffer register to change, which is longer than is the case with the other address registers. The increase is explained by the fact that the *I/O* control unit is remote from the control unit;

(4) delay t_4 which occurs because it takes about 1 μs to wait for a code to be read from the magnetic internal memory.

With the selector channel, the input-output operations are initiated by an instruction from the processor. The control information received by the channel is converted into a sequence of signals which go to the *I/O* control unit. The channel starts the *I/O* control unit to perform the input-output command. The *I/O* control unit generates data transfer requests which are processed by the channel. If the *I/O* control unit generates signals which should be relayed to the processor (for example, an "end of *I/O* operation" signal), the channel will convert these signals to standard format convenient for further use in the processor.

Like multiplexor channels, selector channels are usually built around registers each of which performs a particular function in the reception, storage, checking or modification of control information.

7.7. DATA ENTRY FROM PUNCHED TAPE AND CARDS

Data keyed into punched tape or cards are read into a computer via an input control unit, ICU (Fig. 7.5). Data are read by the devices described in Sec. 6.8. The ICU controls the reader, accepts the data read out and enters them directly into the processor (the arithmetic unit or internal memory).

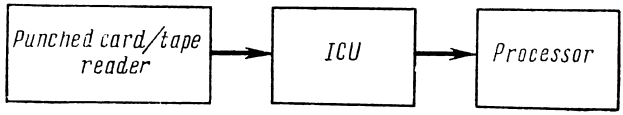

Fig. 7.5. Data entry to a digital computer

Let us see how this is done in the Minsk-32 computer.

Punched tape is read by a FS-1501 reader which operates at a very high speed and can handle five-, six-, seven- and eight-channel tape at 1500 characters/s continuously. At the output of the FS-1501, the data appear as negative rectangular pulses in the same code in which they were punched in.

In block-diagram form, the FS-1501 tape reader is shown in Fig. 7.6. Punched tape is threaded through a drive-and-read unit, *DRU*, which consists essentially of a tape drive, *TD*, and a read head, *RH*. As the tape is advanced past the head, its photocells sense the punched holes and convert them into electric signals

which are then amplified and shaped in amplitude and duration by a signal amplifier-shaper, *SAS*. The tape drive is actuated by an electric motor, *EM*, which also doubles as a fan to cool the heat-dissipating elements. Power for the drive and other circuits is supplied by a power unit, *PU*, which consists of a transformer, rectifier, filter and stabilizer; it furnishes stabilized voltages of +12 V, —6 V, and —30 V.

As already noted, the drive-and-read unit (Fig. 7.7) consists essentially of a tape drive and a read head.

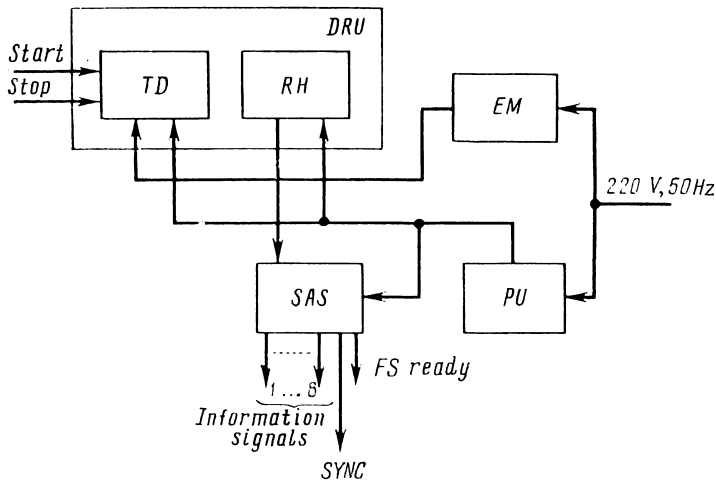

Fig. 7.6. Block diagram of FS-1501 punched-tape reader

The tape drive is a combination of a clutch mechanism and a brake.

The clutch mechanism is a system of two magnetically separated rollers, a capstan, *3*, and a pinch roller, *4*. On one side, the capstan is mounted on the motor shaft, while on the other it is carried by a PTFE bearing press-fitted in a sleeve, *2*. The sleeve is made fast to the core of a clutch solenoid, *1*. The pinch roller is held against the capstan by an angle piece, *6*, over a rubber spacer, *5*.

The brake mechanism has a movable and a stationary part. The stationary part is a U-shaped solenoid, *13*. The movable part is a flat keeper, *11*, held against the U-shaped core by an angle piece, *9*, over rubber spacers, *10*.

Since the tape is permanently held in mechanical contact with the pinch roller and keeper and mechanical displacements are kept to a minimum, the reader operates at a very high speed.

The clutch and brake solenoids are set up on a tape bed, *14*. The pinch roller and the keeper are located in a swivel arm, *7*. The punched tape, *12*, is advanced in the gap between the guide in the tape bed and the swivel arm. The swivel arm simplifies tape threading. During operation, the swivel arm is held down

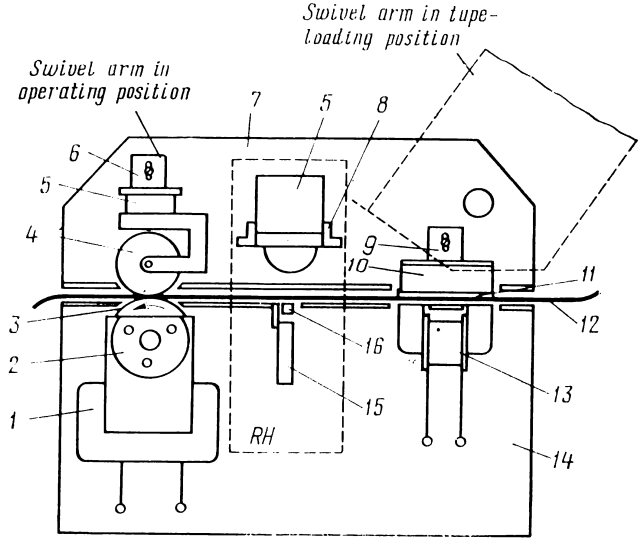

Fig. 7.7. Punched-tape feed

by a spring catch. The tape guide can be adjusted for width with a limit stop which has four fixed positions and can be moved with a knob. The tape bed also gives support to the read head, *RH*, which consists essentially of a light source (a lamp), an objective lens, a prism topped by a cylindrical lens, and a photodiode assembly.

The photodiode assembly, *15*, is located back of the light slit in the tape quide. It has ten photodiodes separated from one another by opaque partitions. Eight photodiodes sense information tracks, the eighth senses sync holes, and the tenth provides a check on the correct threading of tape in the reader. Between the photodiodes and the punched tape is a glass plate, *16*, with rectangular apertures which limit the light beam incident on the

photodiodes in both length and width to a high degree of precision. In the middle of the swivel arm is a movable frame, *8*, which holds a prism, *5*. The prism directs the light beam from the lamp set up sidewise from the tape path onto the tape. The lamp and the objective lens are mounted on the front wall of the reader and are omitted in the figure.

Operation of the FS-1501 reader is controlled by "*Start*" and "*Stop*" signals which are mutually inverted pulses (see Fig. 7.7). After the reader is turned on, tape is threaded in, and the swivel arm is clamped down, a "*FS ready*" signal appears at its output, indicating that the device is ready to read tape.

Until the "*Start*" signal arrives, the clutch solenoid is de-energized, and the brake solenoid is energized. Now the tape is held stationary because the brake force is several times the force that holds the tape against the capstan (which is continually driven by the motor). The narrow light beam is ahead of the next character (the line on tape) so that none of the photodiodes is illuminated. Arrival of the "*Start*" signal de-energizes the brake solenoid and energizes the clutch solenoid so that it attracts the pinch roller, and the latter forces the tape against the capstan. The force with which the capstan grips at the tape increases, the tape is stepped along one position, and the read head senses the next character. For each hole in the tape the read head generates an electric pulse, and this appears on the respective character code wire which corresponds to a 1 in the code combination of the character.

In the case of eight-channel tape, the input control unit may be switched to one of two input modes, "*8 even*" or "*8 odd*". In the "*8 odd*" mode, each line on the tape should contain an odd number of holes. In this way, each character is tested to see if it has been read and transmitted correctly. The pulses generated by the photodiodes are then fed to the signal amplifier-shaper, *SAS*. When the light beam incident on the sync track photodiode is interrupted, a "*Stop*" signal is fed to the FS-1501 reader. With the tape driven at full speed, the device is brought to a complete stop within 1.5 mm.

Apart from control of the FS-1501 reader, the input control unit provides a link to the computer. Accordingly, the unit may be switched to either the "*on-line*" or the "*off-line*" condition.

The "*off-line*" condition is used to test and debug the functional elements of the input control unit. Transfer to the "*off-*

line" condition is accomplished from the control panel, without disconnecting it from the computer.

The "*on-line*" condition is the main form of operation in which the data read from punched tape is entered in the computer.

The unit of information in the exchange between the computer and the input control unit is an eight-bit character (seven information bits and one check bit).

In the input control unit, the character to be entered in the computer is temporarily stored in a character register, and transferred to the computer by a "*Request*" signal generated individually for each character. All bits of a character are transferred to the computer over the eight-code wires simultaneously.

Connection between the input control unit and the computer is set up only for the time interval required to receive the command and to transfer data to the computer.

The input control unit can act in response to two commands, namely "*Enter character*" and "*Enter array*".

The "*Enter character*" command causes one character to be entered in the computer, after which communication between the input control unit and the computer is suspended.

In the case of the "*Enter array*" command, entry of characters to the computer is continued until the computer sends back an "*End of array*" signal, after which communication between the input control unit and the computer is suspended.

The input control unit is synchronized with the cycle time of the computer by clock pulses generated in the computer.

A block diagram of the input control unit is shown in Fig. 7.8. As is seen, it comprises a character register, a check and correct module, a character generator, a control module, a request generator, a synchronizer, a malfunction control module, a power unit, and a control panel.

The character register, *CReg*, receives and temporarily stores a character represented by holes in one line on punched tape.

The check and correct module, *CCM*, tests the character accepted by *CReg* and corrects it, if necessary.

When data are read from five- or seven-channel tape, the data character occupies five or seven bits of *CReg* respectively, and 1 is placed in the eighth check bit if the character read from tape adds up to an "*even*" sum.

When data are read from eight-channel tape, each character is written into bits 1 through 8 of *CReg*. Each character written in *CReg* is subjected to an "*odd*" or an "*even*" parity check, and

if it turns out to be "*even*", it is corrected by inverting its check bit.

The character generator, *CG*, encodes and transfers the character to the computer in a parallel code over eight-code wires.

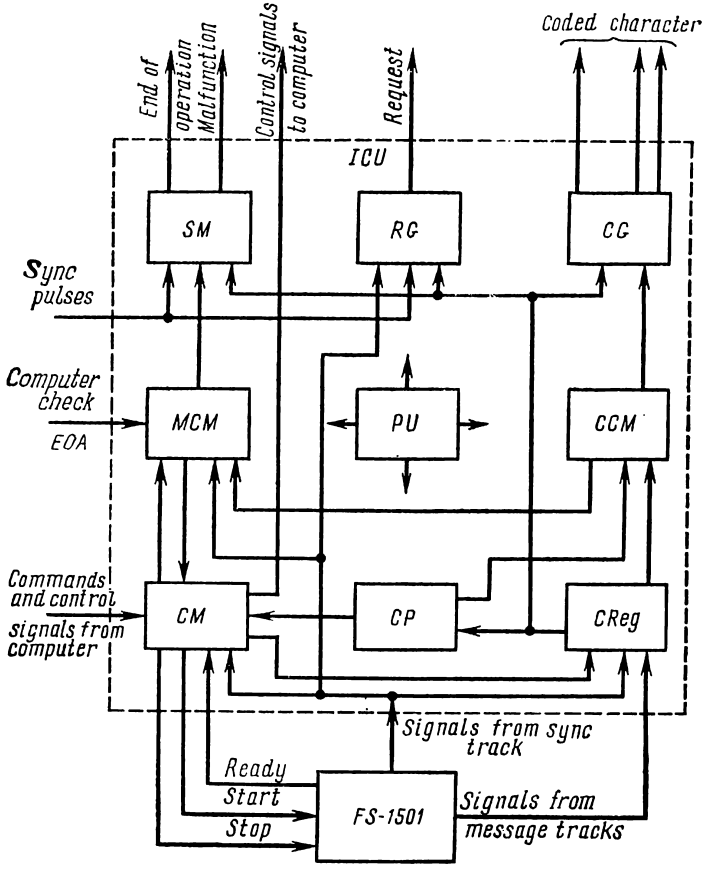

Fig. 7.8. Block diagram of an input control unit

The control module, *CM*, accepts commands and control signals from the computer, starts the input control unit to execute these commands, and relays other signals to the computer. The control module also tests the status of the FS-1501 reader (whether tape has been threaded in, power turned on, and the swivel arm clamped down), starts and stops the tape drive.

The request generator, *RG*, generates a "*Request*" signal synchronized by clock pulses, *CP*, with the internal cycle of the computer.

The synchronizer module, *SM*, generates the "*End of operation*" and "*Malfunction*" signals synchronized with the computer cycle. The "*End of operation*" and "*Malfunction*" signals suspend communication between the input control unit and the computer.

The malfunction control module, *MCM*, senses any malfunctions in the input control unit and in the computer ("*Computer check*") and disconnects the input control unit from the computer.

The control panel, *CP*, provides a choice of three modes of operation: "*off-line*", "*on-line*", and "*tape entry*". The "*off-line*" condition is used to test the input control unit.

The power unit, *PU*, furnishes stabilized voltages of —15 V, —8.5 V and +2.5 V and also alternating voltages of 220 V and 36 V, 50 Hz for the various elements of the input control unit.

Connection between the input control unit and the computer is set up when a reference instruction arrives at the control module. If the input control unit is ready for operation (the "*on-line*" condition has been selected on the control panel and a "*Ready*" signal has come from the FS-1501 reader) and not busy, it proceeds to execute the reference instruction. If the input control unit is busy executing the previous reference instruction, it ignores the reference instruction coming last.

Let us trace how the input control unit executes the "*Enter character*" instruction.

The "*Start-1*" signal coming from the computer switches the input control to the "*Engaged*" condition, while the control module clears the character register and starts the tape drive of the FS-1501 reader. As the tape is advanced past the sensing head, the character lined up with the photodiodes is read out. The character is strobed into the character register by the feed-track signal. The character accepted by the character register is subjected by the check and correct module to an odd parity check (assuming that the tape is read in the "*8 odd*" mode). The corrected character is then fed to the character generator, *CG*.

The trailing edge of the feed-track signal starts the request generator, and the control module generates a signal to stop the tape drive (the "*Stop*" signal). The next clock pulse from the computer causes the request generator to generate a "*Request*" signal. In response to this signal, the computer sends out a "*Start-2*" signal. This signal cancels the "*Request*" signal and

causes the control module to send an "*Entry*" signal to the computer. The computer replies with a "*Transfer*" signal which causes the corrected character to be entered in the computer and the control module to start the synchronizer module. The "*Transfer*" signal from the computer may be followed by an "*End of array*" or "*Computer check*" signal. The control module strobes these signals into the malfunction control module. The next clock pulse from the computer causes the synchronizer module to send to the computer an "*End of operation*" signal which suspends communication between the input control unit and the computer.

The first stage of the "*Enter array*" instruction is executed in much the same way as the "*Enter character*" instruction. The only difference is that after the "*Transfer*" signal has caused the first character to be entered in the computer, the control module clears the character register and starts the tape drive of the FS-1501 reader in order to read the next character on the tape. This sequence is repeated all over again until the malfunction control module receives an "*End of array*" signal from the computer. Because the "*End of array*" signal comes later than the "*Transfer*" signal, the input control unit has time to perform one more entry cycle. During this cycle, the "*Transfer*" signal causes the check module to inhibit a new start of the tape drive and starts the synchronizer module to generate an "*End of operation*" signal.

Should a character be read in error, the check and correct module will generate an "*Even*" signal. In that case, the trailing edge of the feed-track signal causes the malfunction that has occurred in the input control to be registered in the malfunction control module. The "*Transfer*" signal causes the corrected character to be entered in the computer and starts the synchronizer module. The next clock pulse causes a "*Malfunction*" signal to be sent to the computer, and communication between the input control and the computer is suspended.

Should a malfunction occur in the computer during data transfer from the input control, a "*Computer check*" signal will be registered by the malfunction control module. Following the "*Computer check*" signal, the input control unit will enter one more character in the computer and discontinue operation. Neither an "*End of operation*" nor a "*Malfunction*" signal is generated in such a case.

About the same sequence takes place when data are entered in the computer from punched cards, but the input control is

somewnat different. As an example, let us examine the ВУ-600 input device.

The ВУ-600 can read punched cards either continuously or in a start-stop manner. In continuous operation, it handles up to 600 cards/min; in start-stop operation, up to 300 cards/min. Punched cards are read photoelectrically, a column at a time, with all the twelve positions in a column sensed simultaneously.

Cards are fed with their narrow edge first.

The hopper and stacker capacity is 1000 cards each.

The power drive of the ВУ-600 device uses a type АОП-1214, 1400-rpm, 380/220 V, 180-W motor.

The motor is started and stopped, the type of card entry is selected, and operation of the ВУ-600 device is controlled by external control signals.

Should the stacker be overflown or the hopper be left empty, the ВУ-600 will send a "*Pocket*" signal into the external circuit. It also generates synchronizing pulses and character codes.

The ВУ-600 needs for its operation 380/220 V, 50 Hz, three phase, and also direct voltages of —15 V, —8.5 V, and —25 V.

In diagrammatic form, the ВУ-600 device is shown in Fig. 7.9. It may be divided into a transport section, a feed section, and a picker-knife control section.

TRANSPORT SECTION. Rotation of motor *1* is transmitted by pulleys *2* and *3* and V-belt *4* to shaft *5*. The latter mounts gear *6* which meshes with gears *7* and *8* mounted on transport shafts *9* and *10*. Above them are rubber-covered shafts *11* and *12* which are held by springs *13* and friction bearings in contact with, and are actuated by, bare metal shafts *9* and *10*. Shaft *9* carries light-chopper disc *14* on one side of which is photodiode *15* and on the other side, lamp *16*.

FEED SECTION. Rotation of shaft *5* is transmitted by pulleys *17* and *18* and V-belt *19* to shaft *20* which mounts eccentric *21*. The latter causes bellcrank *22* pivoted on shaft *23* to move shaft *24* to and fro. In so doing, shaft *24* actuates movable plate *25*. One arm of the bellcrank is fitted with roller *26* which is held in contact with the eccentric by spring *27*. Also, spring *27* together with limit stop *33* slows down the movable plate in its travel towards the transport shafts. To pivot *24* is hinged tie-rod *28* which is made fast to picker knife *29*. Pivot *23* mounts picker-knife interlock interposer *30*. When pivot *23* rotates clockwise, the interposer is swung into the path of the light beam incident on photodiode *32* from lamp *31*.

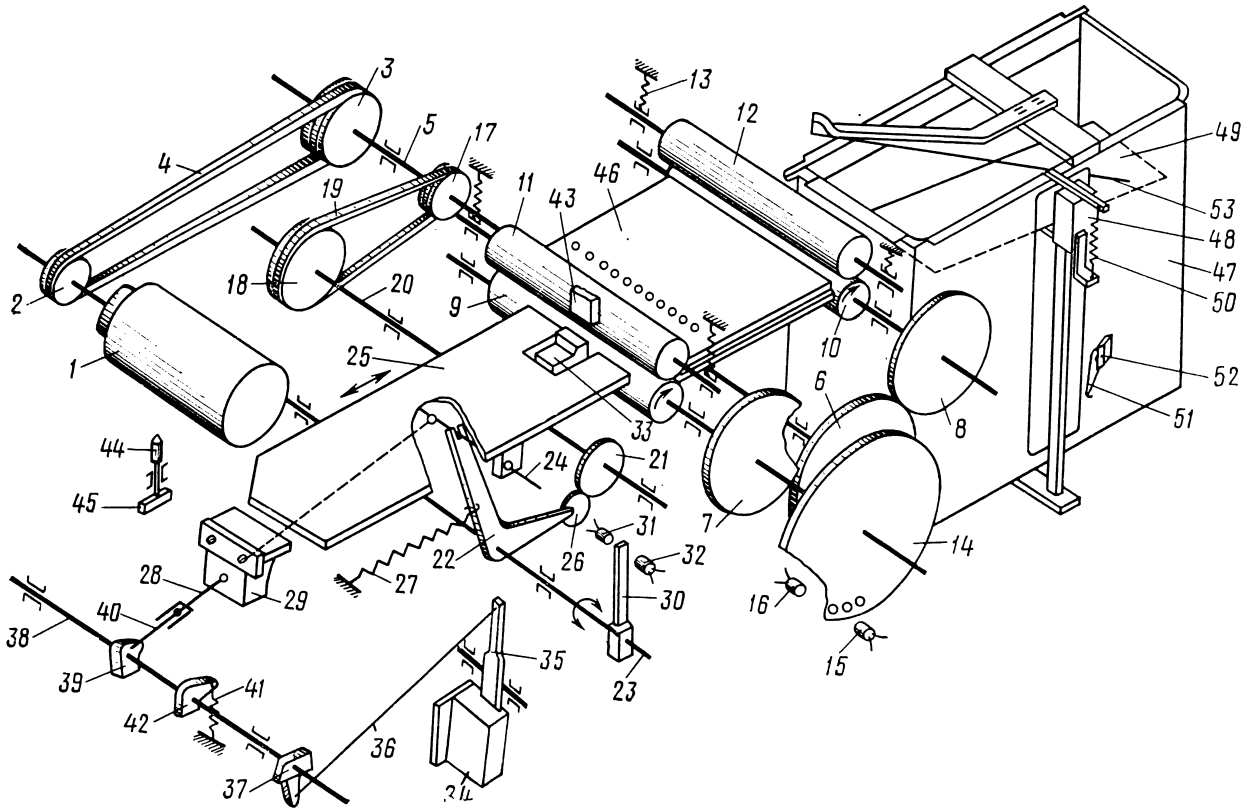

Fig. 7.9. Mechanism of type ВУ-600 input device

PICKER-KNIFE CONTROL SECTION. When an external control signal excites solenoid *34*, armature *35* actuates link *36* and lever *37* to rotate pivot *38* counter-clockwise. In turn, pivot *38* actuates lever *39* and tie-rod *40* inserted into tie-rod *28*, to lift picker-knife *29* to its operating position. Tie-rod *40* is hinged to lever *39*. In its home position, the picker-knife is held by spring *41* which acts through lever *42* mounted on pivot *38*.

A deck of cards is loaded into the hopper so that the lowermost card rests on movable plate *25*. At the bottom, the hopper has a throat formed by limit stop *33* and plate *43* attached to the hopper wall. The throat is only slightly greater in width than the thickness of a card. The loaded deck of cards brings pressure to bear on button *44*, and this closes microswitch *45*. When this happens, a signal is sent into the external circuit that it is proper to start the reader. When the motor starts running, the movable plate travels to and fro, but the cards remain stationary because the force of friction between the cards exceeds that between the lowermost card and the movable plate. At the instant when interposer *30* intercepts the light beam falling from lamp *31* onto photodiode *32*, a signal is generated to notify that it is proper to energize solenoid *34*. If a control signal happens to be applied to solenoid *34* at that instant, picker-knife *29* will be lifted to its operating position, pick the lowermost card as the movable plate travels towards the transport rolls, and push the card through the throat between transport rolls *9* and *11*. After it passes through reading station *46* and a second pair of transport rolls, *10* and *12*, the punched card is dropped in stacker *47*. When the stacker is filled full with cards (that is, when it has collected 1000 cards), carriage *48* fitted with dropping bottom *49* and supported by spring *50* goes down, actuates limit stop *51* to operate microswitch *52*, and the latter stops the reader. On their way into the stacker, the cards are guided by curved strip *53*.

In the start-stop operation, solenoid *34* is de-energized each time a card is fed in. The next card is caused to move into the reading station by another control signal.

In the continuous operation, solenoid *34* remains energized and cards are fed through non-stop.

7.8. OUTPUT CARD AND TAPE PUNCHES

The results turned out by the computer are displayed as holes punched into tape or cards. This is done by output tape and card punches. In principle, these devices do not differ from those in-

tended for the manual preparation of punched tape and cards (see Sec. 6.2).

In fact, some computers use the same punches for both input and output (for example, ПЛ-80 tape punches), or similar devices having a higher speed (such as ПЛ-150 and EC-7022). Connection to the computer and control of operation is accomplished by *output control units,* OCU. Let us see how such a unit operates in the Minsk-32 computer where it controls a ПЛ-80 tape punch.

The OCU has a choice of two operating modes, "*on-line*" and "*off-line*". The "*off-line*" condition is mainly intended for use during the testing and alignment of the OCU. Transition to the "*off-line*" condition is effected from the control panel of the unit without disconnecting it from the computer.

The unit of information in data transfer between the computer and the output control unit is an eight-bit *character* (seven information bits and one check bit). The character to be transferred from the computer is temporarily stored in a character register. Characters are transferred from the computer in response to requests from the OCU. A separate "*Request*" signal is generated for each character. The code bits of a character are transferred from the computer in parallel form over eight code wires.

Connection between the OCU and the computer is set up only for the time interval required to put through instructions and receive data from the computer. The OCU implements two reference instructions, "*Out character*" and "*Out array*". The "*Out character*" instruction causes one character to be outputted from the computer, after which communication between the OCU and the computer is suspended. The "*Out array*" instruction causes characters to be transferred from the computer until an "*End of array*" signal comes from the computer after which the connection is terminated.

Operation of the OCU is synchronized with the computer cycle by clock pulses, CP, generated within the computer.

The OCU can transfer data onto five-, six-, seven- and eight-channel tape. The desired type of tape is selected on the control panel of the OCU. In the case of five-, six- and seven-channel tape, the respective most significant bits of each character are blocked. In the case of eight-channel tape, the OCU may be switched to one of two modes, "*8 even*" or "*8 odd*". In the former case, each line on tape should contain an even number of holes. The seven least significant bits of a character are transferred

onto tape by copying, that is, a hole is punched in tape when there is a 1 in the respective bit position. The characters transferred onto tape are not tested for validity by the OCU.

The OCU is kept in synchronism with the associated ПЛ-80 tape punch by appropriate signals from the latter, such as "*Receive code*", "*Ready*" and "*Start of cycle*".

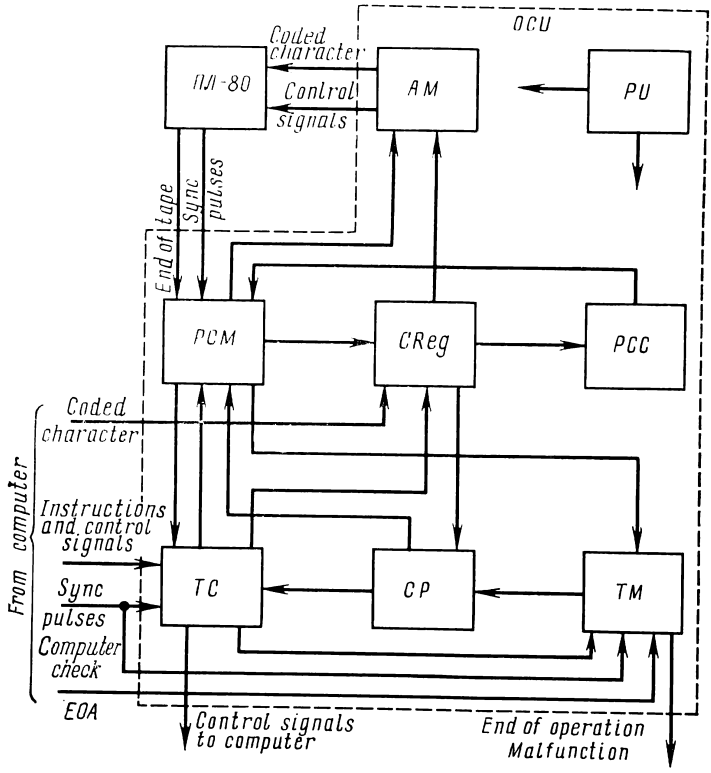

Fig. 7.10. Block diagram of an output control unit

In block-diagram form, the OCU is shown in Fig. 7.10. As is seen, it incorporates a character register, *CReg*, a parity-check circuit, *PCC*, a punch control module, *PCM*, an amplifier module, *AM*, a transfer control circuit, *TC*, a termination module, *TM*, a control panel, *CP*, and a power unit, *PU*.

The character register accepts the character transferred from the computer and stores it for the duration of punching.

The parity-check circuit performs an odd parity check on each character written into the character register.

The punch control circuit maintains the time sequence in which the code bits are transferred from the character register onto tape, starts and stops the drive motor of the ПЛ-80 tape punch, and gates sync pulses coming from the tape punch.

The amplifier module boosts control and code signals in power and passes them onto the tape punch.

The transfer control circuit receives commands and control signals from the computer, starts the OCU to execute these signals, and acknowledges their receipt to the computer. The transfer control circuit gates the code bits, the "*End of array*" (*EOA*) and "*Computer check*" signals from the computer. The termination module senses malfunctions in the OCU and also receives malfunction signals (*Computer check*) during data transfer from the computer, the "*End of operation*" and "*Malfunction*" signals synchronized with the computer cycle. The "*End of operation*" and "*Malfunction*" signals terminate data interchange between the OCU and the computer.

The control panel is used when the OCU is to be tested in the "off-line" condition. It offers the choice of three conditions: "off-line", "on-line", and data output.

The power unit furnishes stabilized voltages of —27 V, —15 V, —8.5 V and +2.5 V and an unstabilized voltage of —40 V for the OCU.

The OCU operates as follows.

Connection between the OCU and the computer is set up the same instant the transfer control circuit receives a reference instruction. If the OCU is ready to operate (the control panel is switched to the "on-line" condition and there is no "*End of tape*" signal from the tape punch) and is not engaged, the OCU proceeds to execute the instruction received. If the OCU is busy executing the previous reference instruction, it ignores the instruction received last.

Let us trace operation of the OCU as it executes the "*Out character*" instruction.

A "*Start-1*" signal switches the OCU to "*Engaged*", and the transfer control circuit clears the character register. The "*Start-1*" signal is then followed by an "*End of transfer*" signal which causes the transfer control circuit to generate a "*Request*" signal. In response to the "*Request*" signal, the computer sends out a "*Start-2*" signal which causes the transfer control unit to send

an "*Out*" signal to the computer, to cancel the "*Request*" signal, and to generate "*Receive character*" and "*Receive EOA and computer check*" signals. The "*Receive character*" signal gates the character code coming from the computer into the character register. The "*Receive EOA and computer check*" signal gates the "*EOA*" and "*Computer check*" signals from the computer. When the last two signals come, the OCU terminates the data transfer, from the computer. The "*Out*" signal causes the computer to put out the character slated for transfer. The character written into the character register is subjected to an odd parity check by the parity-check circuit. The result of the parity check (which may be "*Odd*" if the character has been received correctly, or "*Even*" if it has been received in error) is sensed by the punch control module. The "*End of transfer*" signal sent out by the computer immediately after the character code causes the transfer control circuit to cancel the "*Receive character*" signal. Also, the transfer control circuit cancels the "*Receive EOA and computer check*" signal and generates a "*Start*" signal, but it does so with some delay because the "*EOA*" and "*Computer check*" signals come later than the "*End of transfer*" signal.

The "*Start*" signal causes the punch control module to generate a signal which starts the drive motor of the tape punch, if the motor has not been on already, and gates the "*Receive code*" sync pulses from the tape punch. The leading edge of the pulse which starts the motor disables the reception of the "*Ready*" and "*Start of cycle*" sync pulses by the punch control module. The duration of the disabled state is 0.3 s, necessary for the tape punch motor to come up to speed. In response to the "*Receive code*" signal, the punch control module enables the character written into the character register to be punched on tape. For better reliability, the first character is punched several times over the interval during which the reception of the "*Ready*" and "*Start of cycle*" signals is disabled. At the end of that interval the "*Ready*" signal is gated in, and the punch control circuit stops punching the character, generates an "*End of punch*" signal, clears the character register, allows the tape to be stepped along to the next position, stops gating the "*Receive code*" signal and, with a delay of a few seconds in order to make sure no "*Start*" signal comes again, turns off the tape-punch motor.

The "*End of punch*" signal starts the termination module. The next clock pulse from the computer causes the termination module to send an "*End of operation*" signal to the computer and ter-

minate interchange between the OCU and the computer. The "*Start of cycle*" signal, coming from the tape punch, causes the punch control circuit to stop the tape transport.

The first stage of the "*Out array*" instruction is executed in much the same manner as the '*Out character*" instruction. The only difference is that the "*End of punch*" signal generated after the first character has been transferred onto tape, the transfer control circuit generates a "*Request*" signal which initiates a new transfer cycle to put out the next character. The new transfer cycle commences while the tape is being stepped along as part of the previous cycle. The second and all succeeding characters are only punched once, because the motor remains on, and no signal to disable the reception of the "*Ready*" and "*Start of cycle*" signal is generated.

This sequence of events continues until an "*EOA*" signal comes from the computer. The "*EOA*" signal starts the termination circuit, and this generates an "*End of operation*" signal. No "*Start*" signal can be generated by the transfer control circuit any longer. The character written in the character register remains there.

Should a character transferred from the computer into the character register have an even number of ones, the parity-check circuit will generate an "*Even*" signal which is then put in the punch control module. When a "*Start*" signal comes, the punch control module starts the termination circuit, and this sends a "*Malfunction*" signal to the computer during the next clock period. As a result, interchange between the OCU and the computer is terminated, and the erroneous character is not punched out on tape.

Should a malfunction occur in the computer during an interchange with the OCU, a "*Computer check*" signal will be generated and relayed to the termination circuit. As a result, the character written into the character register is neither checked for parity nor punched out. No "*End of operation*" or "*Malfunction*" signal is generated in such a case.

Output data are transferred onto cards by output card punches. In contrast to card punches used for entry preparation, these card punches have no keyboards and look differently in appearance. As an example, consider the ПЭМ 80 output card punch which operates with 80-column cards.

Prior to operation, a deck of blank cards, *2* (Fig. 7.11), is loaded in the hopper, *1*. As a rule, the deck will fill the hopper to

capacity (700 cards). The path of a card from hopper to stackers may be divided into five steps. During each step one card is fed from the hopper into the card-feed mechanism and the card already on the card bed is advanced a run distance (which is equal to the card width plus the spacing between adjacent cards).

During the first step, the lowermost card is picked by the picker knife from the deck and pushed through the throat towards the first pair of feed rolls, *3*, which grip it and pull along.

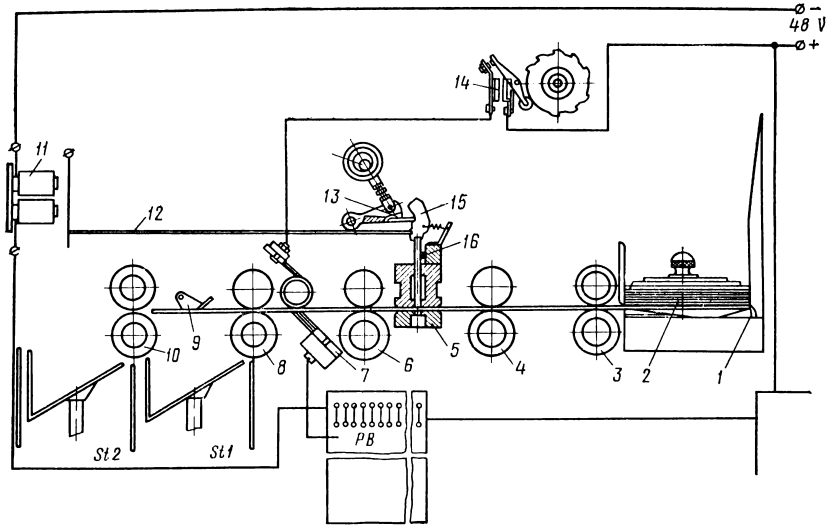

Fig. 7.11. Type ПЭМ-80 output card punch

During the second step, the card keeps moving on and is caught between a second pair of intermittently rotating feed rolls, *4*, which align it with the punch blades in the punch station. The second pair of feed rolls rotate intermittently so that holes can be punched in the card while it is stationary (the card is slightly flexed between the continually rotating rolls *3* and the intermittely rotating rolls *4*).

During the third step, the 80 punch magnets of the punching station punch holes in the card. Each punch magnet, *11*, consists of two coils actuating a single armature. The punch magnets are energized with control pulses from appropriate power amplifiers. When a pulse energizes a punch magnet, its armature is attracted, and interposer arm *12* moves interposer *15* into a position under the end of punch bail *13*. Rotation of an eccentric

shaft causes the punch bail to move to and fro. As the punch bail strikes the interposer, the latter drives the punch pin, *16*, into the die, *5*, thus perforating the card. At that instant, the code magnet is de-energized, while the punch bail, interposer and punch pin are withdrawn to their home positions. Meanwhile, a third pair of intermittently rotating rolls, *6*, advance the card in the reading station, *7*.

During the fourth step, the holes punched during the previous step are sensed in the reading station. This is done by a brush assembly which has 80 brushes arranged in line at right angles to the card path. When no card is placed in the reading station, all brushes touch a contact roller. When a punched card is placed in the reading station, the reading brushes can make contact with the roller only where holes are detected. As a result, a pulse appears in the circuit which runs from the +48 V terminal, via breaker *14*, the "live" brush, the contact roller, to the sensing brush. The pulses are then routed to the computer for a check that the holes have been punched correctly.

During the fifth step, if there is no signal at the sorting station, the card is directed by a guide in between the fifth pair of transport rollers, *10*, rotating continuously and at a high speed, which drop the card in the second stacker *St2*. If there is a control signal fed to the sorting station, the guide will direct the card into the first stacker, *St1*.

The power amplifiers that energize appropriate code magnets are selected by the control unit. All data to be punched into a card are transferred from the computer to the memory module of the control unit in the "*Enter*" mode. Then, in the "*Translate code*" mode, the data are converted from the binary machine code in which the cards are punched into a 12-unit code in which the data are rewritten in the same memory module. However, data are transferred between the computer elements in the form of 8-digit characters, and the locations in the memory module likewise have eight-digit addresses. Because of this each 12-unit character is broken up into two halves and each is placed at an address of its own. This is done so that the data to be punched into the first half of a card (rows 12, 11, 10, 1, 2, and 3) are placed at even addresses in the memory module, and the data to be punched in the second half of the same card (rows 4, 5, 6, 7, 8 and 9) at odd addresses.

When row 12 of a card is aligned in the punching station, the control circuit generates control potentials from which the control

unit derives a "*Position*" signal. This signal switches the device to the "*Punch and check*" condition.

In the "*Punch and check*" condition, data are transferred from the memory module onto the card (the "*Punch*" sub-condition) and the holes are checked to see if they have been perforated in the correct positions (the "*Check*" sub-condition).

At the same time, the "*Position*" signal initiates the operating cycle of the memory module, and the contents of the memory module is read at the zeroth address. Of the data read out, only those are selected which correspond to the position being punched, which is determined by the position counter. If a 1 has been read, it is applied as an "*Enter in IReg*" signal to the least-significant bit flip-flop of the intermediate register, *IReg*, and also as a "*Punch 1*" signal to the parity sum counter in the parity-check unit. Then the next even address is set, and the contents of the intermediate register is shifted one place left. After that, the next cycle is performed. During the second and all succeeding cycles (up to the 16th), operations similar to those of the first cycle take place.

When the 16th even address of the memory module is referenced, an "*End of memory cycle*" signal is generated. This completes the first memory cycle. After the sixteen accesses to the memory module, the data corresponding to position 12 at these even addresses have been transferred to the intermediate register. When the "*Second memory cycle*" and "*Punch enable*" levels are set and a "*Clear punch*" signal is generated, the data stored in the intermediate register are transferred to the punch channel unit to be stored by the amplifiers which control the first 16 code magnets. The intermediate register is then reset. This sequence of events is repeated five times (which constitutes five memory cycles), and the data stored in the intermediate register are distributed among the code-magnet amplifiers by the "*Second memory cycle*" through "*Fifth memory cycle*" levels and the "*Last address*" level generated by the address register.

Holes are punched in response to the "*Punch enable*" signal. Also, all bits punched in a given position are totalled by the check sum counter.

This completes the punch and check cycle for a given position and the device now rests waiting for the next "*Position*" signal to be generated. As a result of this cycle:

(a) for the card being punched: all bits are called up from the like bit positions (positions B) at 80 addresses in the memory

module; a string of bits is collected for punching; all selected bits are totalled by the check sum counter; at each address the check bit is added to the bits in positions B, and the sum is written in the same check bit position;

(b) for the card being checked: the check sum counter adds the sum of unpunched bits in a position to the sum of the bits which should have been punched in the same punching position (during the previous cycle of the card punch).

This sequence of steps (punching and checking) is repeated in each punching position (that is, twelve times) of each card and each time 80 addresses are accessed in the memory module.

The card punch has four contacts which keep watch on the progress of cards along their path. The first contact operates to indicate that the input hopper is filled full. The second contact located near the first pair of feed rolls indicates the exit of cards from the hopper. The third contact operates when a card enters the punching station and is located at the second pair of feed rolls. The stay of a card in the reading station is indicated by the fourth contact.

Whenever a malfunction occurs anywhere along the card path, a "*Malfunction*" signal is sent out to the computer. Among the other signals sent by the card punch to the computer are the "*Ready*" signal which indicates that it is proper to put out data, the "*End of card*" signal, the "*Start of card*" signal, and also the signals generated by the reading station. In turn, the computer generates "*Start*" commands and "*Punch enable*" signals.

When the last card leaves the hopper, or the stackers are filled full, or a malfunction occurs along the card path, the "*Ready*" signal is cancelled, and the card punch stops.

7.9. ALPHANUMERIC PRINTOUT

Further data processing is greatly simplified when output data are printed on paper in man-readable form. This is why a good deal of emphasis has been placed on alphanumeric printers. At present, a variety of printers have been developed and are being used. They may all be divided into two general categories, mechanical (or impact) and nonmechanical (or non-impact).

Mechanical printers are those which make use of some mechanical motion to transfer ink from ribbon to paper. Nonmechanical printers make use of other means for selecting and transferring the characters to be printed.

The class of nonmechanical printers may further be subdivided into three groups:

(1) devices in which characters are transferred on paper by the action of light (photography);

(2) devices in which characters are transferred by the action of electricity in one form or another (dry electrosensitive recording, electrolytic recording, electrophotography, electrostatic recording, etc.);

(3) devices which utilize an electromagnetic field (magnetography).

Photography. With this technique, the character to be printed is illuminated and its image is captured on photographic paper or film. This technique provides a very fast means of producing printed data, especially from a cathode-ray-tube (CRT) display device, and produces high-quality hard copies. Unfortunately, its use has been limited because it involves many steps, needs special illumination and materials.

Dry electrosensitive recording. The dry electrosensitive type of printer uses a record sheet of composite structure. The backing is a pitch black layer of current-conducting graphite-loaded paper. The coating applied by flooding is a dielectric material some 3 to 5 μm thick. The coating is in two layers. The top layer loaded with titania pigment gives the record sheet white colour. The layer next to the backing, known as the reading surface, is lead thiosulphate and serves to enhance image contrast.

Recording paper, in sheet or roll form as may be required, is passed between stylus wires so that they touch the top coating at all times. When a current (signalling) pulse is passed from a stylus into the paper, the coating is burned off, leaving the pitch black backing exposed to produce a permanent black mark.

Among the advantages of the dry electrosensitive printing process are high speed and the high strength of record sheets. Unfortunately, the image is of low contrast because the background is gray, the dielectric coating is fragile, and no duplicates can be made by conventional means.

Electrolytic printing. This technique is not new and has widely been used in facsimile. As with dry electrosensitive recording, the image is produced in daylight and directly (that is, neither development nor fixing is necessary).

In an electrolytic printer, a strip of electrolytic paper (that is, one moistened with an electrolyte) is passed between two stylus wires. When a signalling voltage is applied between the

wires, a current passes through the paper and decomposes the electrolyte. As a result, a chemical reaction takes place, which produces a coloured precipitate, and this forms a mark on the paper.

A very important advantage of electrolytic printing is that it can produce an image in a scale of tonal gradations (or shades of gray). Sometimes it is necessary to make part of a text stand out more clearly against the background. With electrolytic printing this can be done easily, because the image contrast can be enhanced or reduced by increasing or decreasing the amount of current flowing through the paper. Thus, image contrast can be controlled at will by varying the duration of signalling pulses.

Another advantage of electrolytic printing is that it permits reproductions to be made (by, say, light-sensitive duplicating machines), using an electrolytic record as a master. To this should be added the fact that the process is simple to apply and the printing station is fairly simple in design.

Electrophotography. The term electrophotography applies to processes which utilize the photoconductive (inner photoelectric) effect in semiconducting materials and the action of an electrostatic field. As in conventional photography, a light source creates a latent image in the recording medium. The recording medium which is a photoconductive material electrostatically charged prior to exposure is then treated with a finely divided coloured powder of appropriate polarity to develop the latent image. For output printer applications, electrophotography is superior to conventional photography because it needs no wet developing.

The light source of an electrophotographic printer is usually a CRT display device. Unfortunately, CRT display devices suffer from low screen luminance and inadequate resolution along the printed line, and need very high-speed optics. Also, CRT electrophotographic printers are rather slow in action, because the spectra of the CRT and the recording medium differ.

Electrostatic printing. Of all printing processes, this one is the fastest. Basically, it reduces to producing a potential image of characters on the surface of a dielectric material, and developing the image xerographically. The potential image of characters is produced by causing free charge carriers to settle on the dielectric surface, and the free carriers are generated in an air gap between electrodes to which a potential difference is applied sufficient to ionize the air gap.

An electrostatic printer has a tape drive which feeds a strip of paper with a high-resistance coating between an assembly of recording electrodes and a common electrode, a latent-image developing station, and a developed-image fixing station. Starting at the developing station, an electrostatic printer operates in much the same way as an electrographic printer.

It is important that the electronic circuit energizing the recording electrodes should maintain the signalling voltage at a value optimally matched to the air gap existing between the printing electrode and the dielectric coating on the paper. Here, 'optimally' means that the signalling voltage should be such that the corona discharge which charges the paper coating will not change to a spark discharge. If the signalling voltage happens to be below the value matched to a given air gap, no printing action will take place. Conversely, an excessive signalling voltage will produce sparking or even arcing. Accordingly, every effort is made to keep the signalling voltage the same for all recording electrodes and to maintain a constant air gap all along the stylus assembly.

Magnetography. This process involves the use of magnetic materials. Latent images are recorded on a magnetic medium and made visible by inking with a powder which clings to the charged portions of the medium. Then the magnetic medium is pressed against a strip of paper, the powder is carried away and secured to the paper in some way. Since the latent magnetic image is retained on the medium, it may be used repeatedly to produce many paper copies, as already explained.

Magnetographic printing can considerably be simplified through the use of special magnetic paper. As in electrostatic printing, it is possible to produce tonal gradations (shades of gray) in a magnetographic image by varying the diameters of marking dots.

Mechanical printers still remain predominant output equipment with general-purpose computers. This is so not because they are perfect, but because nonmechanical printers are sometimes inadequate in performance.

The alphanumeric output devices usually built into users' terminals and control engineers' consoles are the typewriters and teleprinters described in Chap. 6. However, they print at low speeds (10-15 characters/s), and in the latest makes of computers preference is given to alphanumeric printers of special design.

The basic specifications of some Soviet-made alphanumeric printers are listed in Table 7.1.

Table 7.1

Particulars	Type designation			
	EC-7030	EC-7032	EC-7033	АЦПУ-128-2М-3М
Printing speed	650-890 lines/min	900 lines/min	1100 and 550 lines/min	360-440 lines/min
Characters per line	128	128	120, 128, 160	128
Capacity of buffer storage	128 bytes	128 bytes	160 bytes	128 bytes
Power consumption	1.5 kVA	2 kVA	3.5 kVA	1.5 kVA

Most often, alphanumeric printers are of the multiple-wheel on-the-fly types. They have a high printing speed (1000 characters/s and more) because they print a line at a time (which is why they are called line-a-time printers). The number of print wheels is the same as that of characters in a line. A printing cycle for each line is carried out in three steps: the print wheels are positioned, the desired characters are imaged on paper, and the paper is stepped along to the next printing position.

As an example, consider operation of the Soviet-made АЦПУ-128 alphanumeric printer (the numeral '128' indicates that each line has 128 printing positions). It is intended to present output data in man-readable form on paper fed from a roll and folded.

The heart of the on-the-fly printer is the printing action using continually rotating type wheels. The action for each printing position (Fig. 7.12) consists of a hammer solenoid, *2*, an arm, *1*, a printing hammer, *4*, and a type wheel, *7*, on which 78 raised characters are arranged all the way around the circumference.

The 128 hammer solenoids each of which causes a particular character to be printed in a line are built as self-contained assembly. The hammers are arranged in a row (each opposite a type wheel of its own). The type wheels are separated from the hammer tips by an air gap 2 mm wide to pass paper, *5*, and ink ribbon, *6*. The solenoid assembly is built into a housing the side walls of which have guides so that the solenoid assembly can

be made fast to the paper-feed unit. The guides are designed so that the hammer tips can be moved parallel with and at right angles to the type-wheel shaft and relative to the type wheels themselves for adjustment purposes.

Printing takes place as follows. The type-wheel shaft is continually driven by an electric motor via a V-belt and reduction gear. At the instant when the middle of the character to be printed crosses the centre line of the hammer, a signal from the computer energizes the hammer solenoid, and it attracts the arm which strikes the hammer. At first, the hammer moves together with the arm. After the clearance between the solenoid core and arm is taken up, the hammer keeps moving by inertia until it strikes the type wheel. In doing so, it drives the paper and ink ribbon against the type wheel, and the character is thus printed. After that, a spring, *3*, causes the hammer to retract.

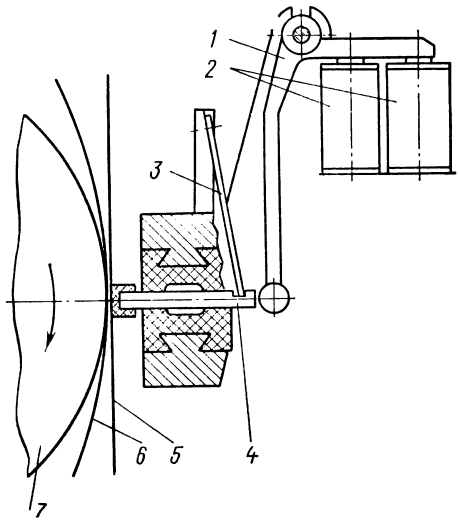

Fig. 7.12. Character printing action

In greater detail, the printing action of the АЦПУ-128 on-the-fly printer is shown in Fig. 7.13. As is seen, rotation of motor *1* is transmitted by V-belt *2* and reduction gears *4*, *6* and *7* to type-wheel shaft *5*. The shaft is a tube with two trunnions carried in bearings which are installed in the walls of the enclosure. The 128 type wheels are keyed to the shaft. On the right-hand end of the shaft is keyed a gear, *19*, which meshes with another gear, *18*, on shaft *20* of a double-thread reduction worm which in turn actuates worm wheel *13*, and this transmits rotation via a pair of gears *14*, *15* and *17* to electromagnetic clutches *16* and *21* which are part of the ink ribbon drive.

When clutches *16* and *21* are energized, coils *9* and *22* mounted on two taper centres and carrying the ink ribbon start rotating. The two coils are identical and rotate at the same speed. The direction in which the coils and, as a consequence, the ink ribbon rotate can be reversed with two microswitches. Rotation of gear

19 is also transmitted via gears *18*, *10*, *11* and *12* to the spacing mechanism.

The spacing mechanism advances the paper one space as a line is printed and eight spaces per revolution of the type wheels. It consists of a multi-plate electromagnetic friction clutch and a ratchet mechanism, *8*, to keep the paper precisely on the desired

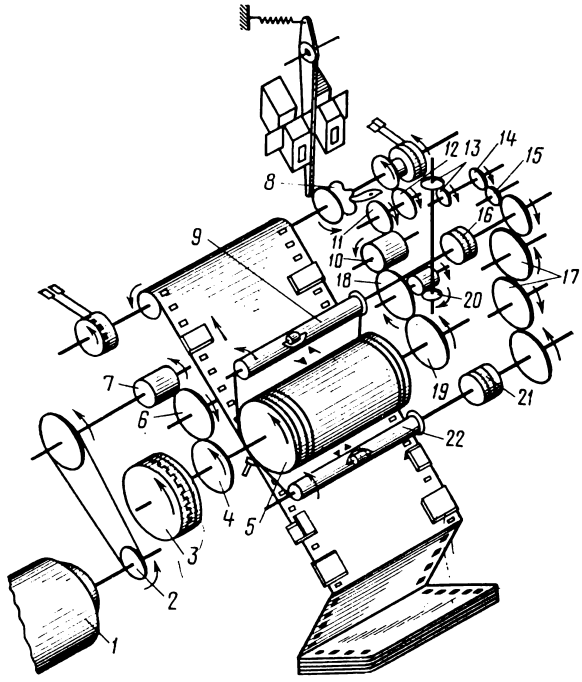

Fig. 7.13. Printing action of an alphanumeric printer

line. The ratchet mechanism has two ratchet wheels and pawls (in the figure only one wheel and pawl are shown), one for forward and the other for reverse motion.

When the clutch engages, rotation of gear *12* is transmitted first to the drive, then to the driven clutch plates mounted on the paper-feed roll. The roll is brought to a stop at the desired line as the respective pawl engages its ratchet wheel under the action of a solenoid. When this happens, the electromagnetic clutch disengages.

The printer is synchronized with the computer by 78 position pulses supplied by an induction generator, *3* (96 pulses for the АЦПУ-128-3М printer).

The 128 characters to be printed on a single line are transferred from the computer into the control unit of the printer, where they are entered via the character register, *CReg*, in the memory, *M1* (Fig. 7.14). In the character register, each character

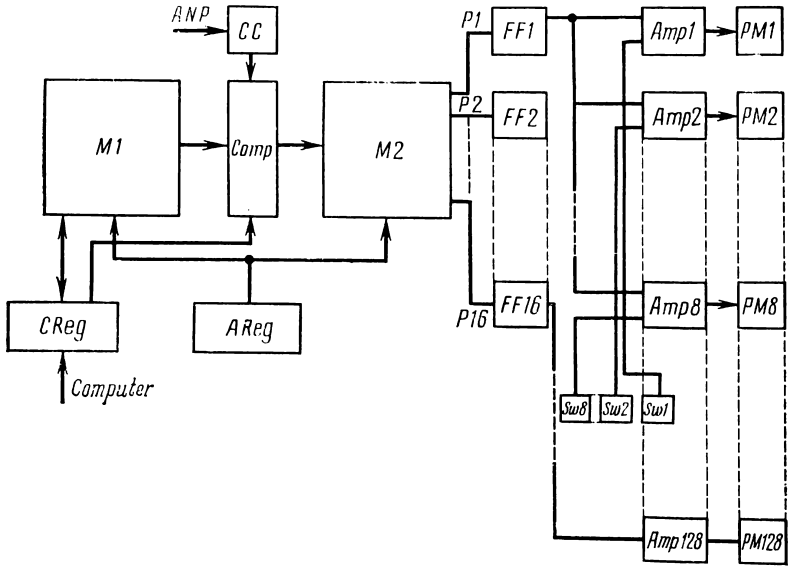

Fig. 7.14. Block diagram of an alphanumeric printer control unit

(code combination) is subjected to an odd parity check. The memory *M1* can store 128 eight-bit code combinations. The combinations are steered into the memory by an eight-bit address register, *AReg*. After the 128 characters have filled the memory full, transfer from the computer is discontinued, and the printer and its control unit switch to an internal cycle.

The first step in the internal cycle is "*Analysis*". During this step, a character (code combination) is transferred from the generator of the type-wheel assembly into a code counter, *CC*. The data stored in *M1*, starting with the first character, are sequentially compared via *CReg* with the contents of the code counter by a comparison circuit, *Comp*. The addresses of the combinations that match the contents of the code counter are stored in the

respective positions in the second memory, *M2*, which is an eight-by-sixteen core matrix. During this step, both memory modules, *M1* and *M2*, are controlled by the address register. After the last, 128th, character stored in *M1* has been compared, the "*Analysis*" phase is terminated and the "*Print*" phase is commenced.

The "*Print*" phase occupies eight clock periods. During each clock period, data stored in the cores threaded by a vertical wire is transferred from *M2* over the sixteen horizontal channels, *P1* through *P16*. For each vertical wire, a timing pulse of its own is generated. The data retrieved during each clock period are entered into the sixteen-bit register made of flip-flops *FF1* through *FF16*, which is cleared prior to every clock pulse.

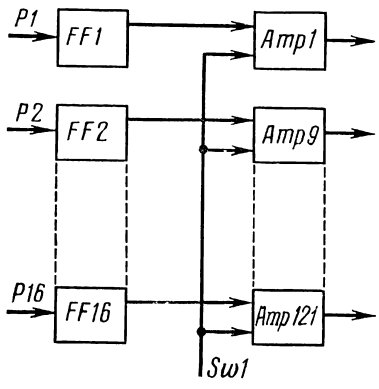

Fig. 7.15. Print control by an alphanumeric printer control unit

Each flip-flop serves eight print magnets, *PM*, in turn via the respective amplifiers, *Amp1* through *Amp128*. Each flip-flop is connected to one of the eight amplifiers by switching pulses coming from the control unit over eight channels. With each clock pulse is associated a switching pulse, *Sw1* through *Sw8*. For example, during the first clock period pulse *Sw1* (Fig. 7.15) causes characters to be printed in sixteen positions (1, 9, ..., 121) of a line according to the contents of the sixteen-bit register. This is done by the solenoids whose amplifiers have at their inputs both the switching pulse *Sw1* and "1" signals from the flip-flops. Each solenoid amplifier consists of an AND gate and a power amplifier.

During these eight clock periods, the character that has been coded in the code counter (see Fig. 7.14) is printed in the respective positions of the line. After that, the next character is coded in the code counter, and the events just described are repeated all over again. A complete line is printed during one revolution of the type-wheel shaft.

Printout starts with the character whose code is fed by the generator of the type-wheel assembly to the code counter at the commencement of the internal cycle and stops when the same character appears in the code counter.

7.10. TERMINAL MULTIPLEXOR

The term *terminal multiplexor* defines a facility which links a computer to communications channels. A multiplexor is connected to the multiplexor channel of the computer via an I/O interface and enables the computer centre to give service to any desired number of remote terminals. Apart from connection of a computer to communications links, the terminal multiplexor can automatically set up connections to users, convert data from one code to another, switch single-address and multi-address messages, etc.

As a link between a computer and communications channels, the terminal multiplexor should be built to match their capabilities. For example, the type 8401 terminal multiplexor used within the ES EVM unified computer structure enables a single computer structure to be connected to 31 communications links and I/O operations to be performed at the rate of 50 to 1200 bits/s. The Minsk-1560 terminal multiplexor can attach up to 32 telegraph lines to a Minsk-32 computer.

Consider operation of a terminal multiplexor, using the Minsk-1560 as an example.

This terminal multiplexor allows a computer to be connected to both private (rented or leased) and switched telephone and telegraph lines. Data can be transferred over the 32 lines at the same time without any interference.

In operation over telegraph lines, the terminals are teleprinters (such as PTA-6 or T-63) and other devices equipped with terminal adaptors capable of operation in international telegraph alphabet No. 2 at 50 bits/s. As many as four telegraph channels can be replaced with four telephone channels, using a suitable communication adapter, or modem (see Sec. 5.6). In such a case, the sending speed will be determined by the data communication equipment.

The users served by such a system can remotely enter their data in the computer, send in request for reference information, and receive results of computations. For entry of data into the computer, a telegraph-network user should first set up a connection. This is done automatically as he dials the computer centre and exchanges identity characters with the Minsk-1560.

Where data entry is over telegraph lines, the amount of data is limited by the size of the input field allocated in the core memory of the computer, which is done by the programmer as he

compiles a working program in each particular case. No identity characters need be interchanged in operation over telephone lines.

Data output from the computer to users is software-controlled. Connections with the users of a leased telegraph network are set up automatically, with the users of a switched telegraph network via a service teleprinter and a dial unit and with the users of a telephone network via a telephone set.

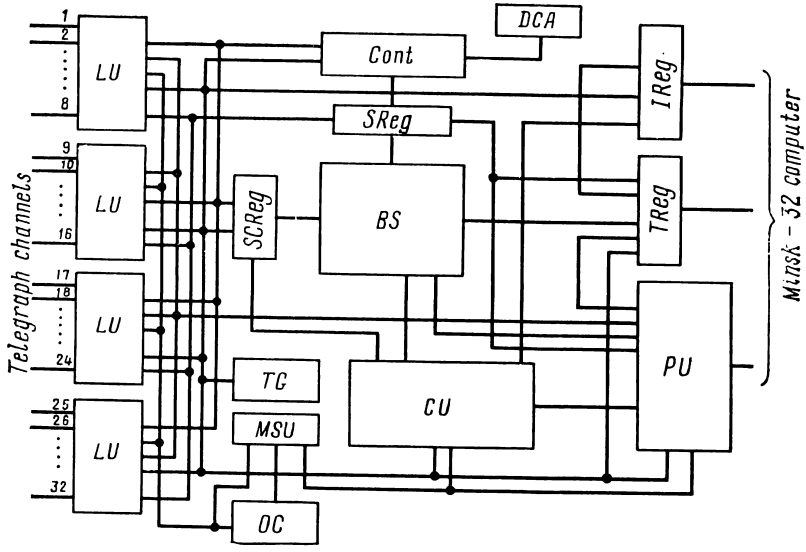

Fig. 7.16. Block diagram of type Minsk-1560 terminal multiplexor

The service teleprinter may also be used for data transfer to and from the computer (abridged service), and also to monitor and exchange data with any of the telegraph channels.

The Minsk-1560 terminal multiplexor has no provisions for error control; this is done by software at the computer end of the line, and the user involved may then be informed about the outcome of the validity check.

In block-diagram form, the Minsk-1560 terminal multiplexor is shown in Fig. 7.16.

Each line unit, *LU*, performs the following functions:

it automatically sets up a connection with eight telegraph channels;

it converts telegraph signals into standard pulses and back;

it stores one bit of information received from a channel or intended for output to a user;

it protects the circuit against false operation in the presence of interference and noise in the channel;

it can guard any of the eight channels connected to it;

it clears the channel when the user rings off.

The control unit, CU:

generates timing pulses independently and in response to clock pulses from the associated computer;

connects the service teleprinter to the computer and to any of the connected channels for test purposes or as a user's set;

enables an external user to signal the operator;

provides a means for accessing the buffer storage *BS*;

stores the address of the channel over which a request is submitted for a data-transfer cycle;

enables identity characters to be exchanged after a connection has been set up and prior to data entry;

can hold all channels;

controls the various devices in off-line operation;

generates an autoanswer from the computer.

The buffer storage, *BS*, can store 128 eight-bit words and is divided into 32 areas (as many as there are communication channels) each of which consists of four eight-bit locations or memory cells. Thus, each channel is allocated four memory cells: the first cell stores service characters, the second, status indicators (such as "*Ready*", "*Engaged*", "*Malfunction*", etc.), the third saves the contents of the shift register, *SReg*, when a serial code is converted to a parallel one or vice versa; and the fourth cell stores temporarily a character received from or intended for transmission into a channel.

The service-character register, *SCReg*, stores internally and puts out service levels in operation over a given channel.

The eight-bit shift register, *SReg*, interchanges data with telegraph channels bit by bit, and with the line or terminal adaptor in a parallel code.

The terminal multiplexor includes a special interrupt register, *IReg* so that "*Request I/O*" signals can be generated and operation of the multiplexor can be controlled during a data-transfer cycle.

Characters are transferred to and from the computer and the information (instructions to examine pointers, reference instructions) coming over the character wires is decoded by a transfer

register, *TReg*. Also, the transfer register applies an odd parity check to the code combination received from the computer.

The pointer unit, *PU*, is essential for control of the *I/O* program and also feeds interrupt signals to the computer ("*I/O end of operation*", "*Interrupt request*", "*I/O malfunction*") and decodes the status pointers that have initiated the interrupt, so as to determine in which direction the *I/O* program is to operate.

For data entry over a telegraph channel, the user sets up a call from his teleprinter and presses the "*Figures*" and "*Who are you?*" keys on his keyboard. If these signals are received correctly, the Minsk-1560 generates and releases an answer in the form of a "=" sign. On receiving this character, the user should press the "=" key on his keyboard, which signifies that he requests clearance to enter data. As a result, the "*Entry request*" flag is set in the terminal multiplexor, and an "*Interrupt request*" is sent to the computer in order to interrupt the main program. After his request has been analyzed and there is an unoccupied "input field" in the core memory, the user is sent a "=" sign, which indicates that it is proper for him to enter his data. The transmitted data are fed serially to a line unit where the telegraph signals are converted to pulses and one bit of information is stored. Then the data are fed bit by bit via the shift register to the buffer memory where the serial code is converted to a parallel one and character strings are formed.

Data are written into the buffer memory bit by bit as follows: the contents of the third memory cell of the buffer memory is transferred into the shift register, *SReg*, the contents of the shift register are shifted one position, the next bit of the code combination coming from the channel is placed in the least significant position, and the contents of the shift register are transferred to the buffer memory. In this way, the third cell of the buffer memory is filled bit by bit until a 1 (the start pulse) appears in the sixth bit of the shift register (the five less significant bits hold an information combination in international telegraph code No. 2). Then a check bit is added, and the contents of the shift register are written in the fourth buffer memory cell of the channel involved, where it is stored until entry into the computer.

The speed of the shift register and buffer memory is such that all the 32 channels can be served within a single cycle time of the multiplexor by receiving code combinations from all channels bit by bit.

Data are received and written into the buffer memory during an operating cycle of the channel. The operating cycles of channels are arranged sequentially and form an operating cycle of the terminal multiplexor.

For each channel, the timing generator, *TG,* generates nine timing pulses in response to clock pulses from the computer. Every ninth timing pulse changes the number of the channel being served. The servicing time of a channel is determined by the duration of the level representing the channel No.

Data are entered in the computer during a transfer cycle organized by requests from the terminal multiplexor, and initiated by a signal from the computer. In response, the multiplexor sends out an "*Input*" signal. The same signal is transmitted to appropriate multiplexor elements necessary for data transfer. During the transfer cycle, the code combination is moved from the fourth buffer memory cell via the transfer register to an appropriate address in the core memory of the computer. Input is terminated when an "*End of message*" flag occurs. This flag causes an interrupt signal to be sent to the computer, and the "*Engaged*" flag to be cleared in the multiplexor for the channel involved. The computer switches to processing the received message, and the channel is reset at the end of the internal cycle.

Data entry may be accompanied by malfunctions. This is why each channel has a "*Malfunction*" flag and a channel-malfunction control unit, *MCU*, so that such conditions can be properly analysed, indicated and cleared. When an input malfunction occurs, the "*Malfunction*" flag is set to "1", and a "*I/O malfunction*" signal is fed to the computer, causing it to switch to the "*Malfunction service* (*MS*) *routine*". Input continues, but data are not transferred to the computer. After the "*End of message*" combination is decoded, the multiplexor generates a "*C*" character which signals the user that a malfunction has occurred in the course of entry and that he should repeat his message.

Data output from the computer to users over telegraph channels commences after a connection has been set up. With private channels, connections are set up automatically by an "*Output*" command. With switched telegraph channels, connections are set up by an operator at the computer centre to whom the computer makes known the No. of the channel to use and the codeword of the wanted user. Using his console, *OC*, the operator proceeds as ordered and disconnects the service teleprinter as soon as the call is set up; this automatically connects the called

channel to the computer. In other respects, the procedure for data output is the same for private and switched channels.

After a connection has been set up, the computer receives an "*I/O free*" signal if the channel is not engaged and ready to operate, while in the multiplexor the "*Output*", "*Connection with computer*" and "*Engaged*" flags are set to "1". As soon as the "*Output*" flag is set to "1" and it is proved that the fourth buffer memory cell and the transfer register are not occupied, the multiplexor is ready to handle data coming from the computer.

After each combination, an "*End of transmission*" signal comes from the computer. This signal initiates an odd parity check on the received combination. After that, the shift register and the third buffer memory cell convert it into a serial code combination to which one more bit containing a "1" is added to form a "*Stop*" signal, and the combination thus obtained is sent to the line unit.

The line unit converts code combinations into telegraph signals. The last information combination is the "*End of message*". After the "*End of message*" is received, the computer generates an "*End of array*" signal which causes the multiplexor to clear the "*Engaged*" flag, and the line unit is sent a "*Ring off*" signal to break up the connection in the case of a private channel (in the case of a switched telegraph network, this is done by the user). At the same time, an interrupt signal is sent to the computer which recognizes it as an indication that the output cycle is complete.

For transfer of data between the computer and users over telephone channels, a communication adaptor (see Sec. 5.6) should be inserted between the Minsk-1560 multiplexor and the channels.

After the communication adaptor is switched to the "data transfer" mode, a special signal comes to the terminal multiplexor via a controller, *Cont*, and the multiplexor is made ready to receive data. Once the buffer memory is brought to readiness, data begin to come in, bit by bit, in a parallel code to the shift register.

When the first character is entered, a request for entry is submitted to the computer. Following this request, the computer interrupts its main program and enables entry. An access instruction is sent to the multiplexor, and the data begin to be transferred to the computer via the transfer register.

Data can be output from the computer to a user only after a proper connection has been set up. This is done by the operator using a telephone set. After the call has been put through, the operator presses the "*Transmit*" and "*Start*" keys on the data communication adaptor.

On reaching the multiplexor, the "*Transmit*" signal sets the "*Ready*" flag to a "1" state for the channel over which data are to be relayed. The "*Start*" signal causes the computer to interrupt the main program, and data begin to be output via the multiplexor and the data communication equipment to the user.

CHAPTER 8

ANALOG-TO-DIGITAL CONVERSION

The term analog-to-digital conversion applies to transformation that continuous, or *analog*, quantities or measurements undergo as they are converted into digital numbers.

This chapter will take up mechanical and electric analog signals. Accordingly, we shall examine devices which convert mechanical displacements (angular or linear) into digital numbers, and devices which convert voltage and current into digital numbers.

Position-to-digital converters may be divided into three major classes according to the methods they use: (a) incremental (pattern) converters; (b) total-value converters; and (c) indirect converters involving an intermediate conversion of position to time.

In an incremental converter, analog-to-digital conversion takes place as follows. The continuous quantity being measured is broken up in a series of discrete increments (quanta), Δp, and each increment is represented by a 1 in a digital code. The converter generates a 1 each time the position is changed by this increment, and by summing these incremental signals one obtains a digital representation of the position, or rather of the change $(p_2 - p_1)$ in the position from the initial, p_1, to the final one, p_2, over the conversion interval. Incremental converters are fairly simple, but suffer from a major disadvantage: should an error in counting occur, such as a loss of a pulse, the final error will be proportional to the total number of pulses lost — a condition known as a cumulative error. Another limitation of incremental converters is that the conversion is concerned with an incremental change in the variable, and not with its absolute value. With this technique, the initial position, p_1, is coded as zero.

A total-value converter gives an entirely new reading for each position independent of any other position; that is, each position is assigned a particular digital code combination.

In an indirect converter, the conversion of a position to a digital representation involves an intermediate conversion to a

time interval. The time interval is then filled with precisely timed pulses, and their number is counted. The pulse count is the final result of the conversion. This technique is markedly distinct from the incremental method. Firstly, the absolute value, not an increment, of a variable is converted because a certain time interval is assigned to each value of the variable. As the variable changes continually, so does the time interval. Secondly, the time interval is determined several times in such a manner that during the interval of a continuous change in the variable by unity a digital representation of its absolute value can be obtained. Sometimes, A/D converters may use a combination or combinations of the various techniques.

8.1. A/D CONVERSION ERRORS

Consider the errors associated with an A/D converter, or digitizer, which codes a continuous variable into a digital form by sampling and quantizing. It is to be noted that our interest will be in how closely discrete values approach the original continuous quantity (errors in the latter are ignored), rather than in the errors associated with the recovery of the original function from its discrete values. That is, our concern will be with the accuracy of a digitizer proper. We shall base ourselves on observations applicable to a large group (or population) of such digitizers, and not on a particular device intended to digitize a particular variable obeying a specific law. This is why the matter will be approached statistically.

According to their sources, the errors associated with digitizers may be classed into those caused by sampling and quantizing and those stemming from the digitizer itself. The former may be assessed in terms of the rms quantization error, or quantization standard deviation, σ_q. The latter may be assessed in terms of the rms instrument error, or instrument standard deviation, σ_i.

The errors associated with sampling and quantizing have already been discussed in an earlier section. Here, we shall only deal with those arising from amplitude quantization, or quantizing proper, σ_{qq}. In principle, sampling may likewise lead to errors, σ_{qs}, because each sample occupies a finite time interval during which the continuous variable being measured may change by some incremental value. For a digitizer to operate normally, the average sampling time, t_{av}, should be chosen such that the

incremental change in the measured quantity be less than the quantizing step, that is, $t_{av} < t = \Delta p/p$.

The instrument errors include those caused by errors in manufacture and assembly, deformations due to force and temperature, and wear-out with service; noise in the associated electronic gear, etc. In one way or another, these errors affect the accuracy with which a continuous function can be quantized and sampled. Therefore, the effect of all primary errors of a digitizer should preferably be assessed as a fraction of a quantum.

The overall error of a digitizer may be expressed in terms of the overall standard deviation:

$$\sigma_d = \sqrt{\sigma_q^2 + \sigma_i^2}$$

The values for σ_q and σ_i should be chosen individually in each particular case. The overriding requirement, however, is the final accuracy of the digitizer (as a rule, σ_q must be greater than σ_i).

8.2. INCREMENTAL CODE-WHEEL CONVERTERS

A sketch of a simple incremental converter using a code wheel is shown in Fig. 8.1. The code wheel, *W*, is divided into conducting and nonconducting sectors and mounted on the shaft whose angular position is to be converted into a number. As the code

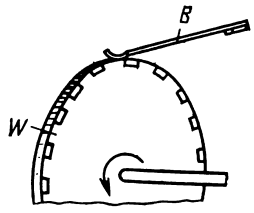

Fig. 8.1. Incremental A/D converter

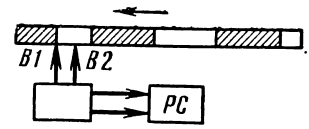

Fig. 8.2. Incremental A/D converter with a reversal feature

wheel rotates, its sectors come in turn under a sensing (or reading) brush, *B*. If the code wheel — brush circuit be connected to a supply source, it will, or will not, carry a current according as a conducting or a nonconducting sector comes under the brush. The number of current interruptions registered by a counter will give a digital representation of the angle through which the shaft has turned, that is, of the shaft position.

The accuracy of this type of digitizer depends on the number of conducting and nonconducting sectors on the code wheel. This in turn depends on the size of the code wheel and the practically attainable width of a sector.

A major drawback of this type of converter is that it has no ways of telling when the direction of shaft rotation is reversed. This drawback is avoided by provision of two brushes, *B1* and *B2*, spaced half the sector width apart (Fig. 8.2), a circuit which senses the direction of rotation by the order in which the signals come from the brushes, and an add-subtract pulse counter, *PC*. When the shaft is rotating, say, counter-clockwise, the pulses are added to those stored previously; when the shaft is rotating in the opposite direction, these pulses are subtracted from the previous count. The logic of the sense determining circuit depends on the manner in which shaft rotation is digitized.

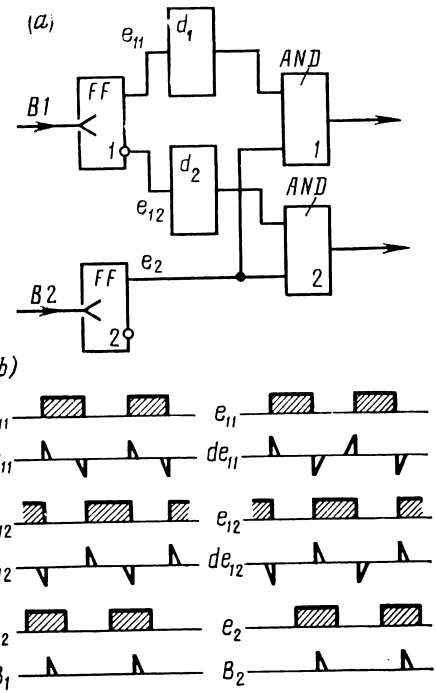

Fig. 8.3. Sense detector
(*a*) block diagram; (*b*) timing diagrams

In the case of time gating, use may be made of the circuit shown in Fig. 8.3*a*. Signals from the brushes *B1* and *B2* are applied to flip-flops, *FF1* and *FF2*. The voltages e_{11} and e_{12} taken from the arms of the flip-flop *FF1* code the number in its true representation and 1's complement form, respectively. These voltages are differentiated by networks *d1* and *d2*; the resultant pulses are applied to AND gates, *AND1* and *AND2*. The AND gates pass only positive pulses when their second inputs accept the positive (true-representation) voltage e_2 from the flip-flop *FF2*. Output pulses appear at the output of the *AND1* gate or the *AND2* gate, according as the flip-flop *FF1* or *FF2* changes state first, which in turn depends on the sense of change in the measured variable. Timing diagrams for both cases appear in

Fig. 8.3*b*. The add-subtract counter adds pulses coming from the *AND1* gate and subtracts those coming from the *AND2* gate.

The incremental digitizer discussed above uses the brush (or electric-contact) reading technique. However, there is nothing to prevent it from using other suitable techniques (an especially high resolution can be obtained with the aid of optical gratings).

As already noted, a major disadvantage of incremental digitizers is the risk of a cumulative error due to a loss of pulses. Also, they can only digitize incremental changes in, and not a continuous variable itself. Because of this, incremental digitizers have found a limited use.

8.3. TOTAL-VALUE A/D CONVERTERS

A total-value A/D converter directly reads a code combination for each quantized level of an analog quantity. This form of converter is most often used to digitize shaft position. The principal reason for this application is that each code combination can be read independently of the previous one. That is, the shaft position can be determined at any sampling instant and no cumulative error arises. Should an error occur in a particular reading, it will in no way affect the next one.

Total-value converters may be divided into code-wheel types, matrix types, and phase-shift types.

Code-wheel converters, as their name implies, use code wheels in order to digitize an analog variable. Accordingly, matrix-type converters do their job with the aid of code matrices or decoders.

Phase-shift converters ordinarily employ a resolver and a frequency multiplier.

Binary code wheels. A *code wheel* assigns to each shaft position a certain definite code combination, that is, a string of characters selected to a specific law. According to the number system adopted, code wheels may be binary, ternary, etc.

In binary coding, each shaft position is represented by a binary number $x_n x_{n-1} \ldots x_1$. If the range, $0 - \varphi_{max}$, of the variable to be converted is known and the least increment in it, φ_b, is chosen such that it satisfies the requirement for accuracy, on the one hand, and the ratio φ_{max}/φ_b is a multiple of some power of two (which is necessary for ease of mechanization), then the

output binary number will have

$$n = \log_2 (\varphi_{max}/\varphi_b)$$

digit positions.

A binary code mask or wheel is constructed by dividing its field into linear or concentric tracks (according as linear or angular motion is to be digitized) and into columns or sectors, respectively. A five-digit binary code wheel is shown in Fig. 8.4 where a 0 is coded as a dark segment and a 1 as a light segment. The reading elements (to be discussed later) are arranged

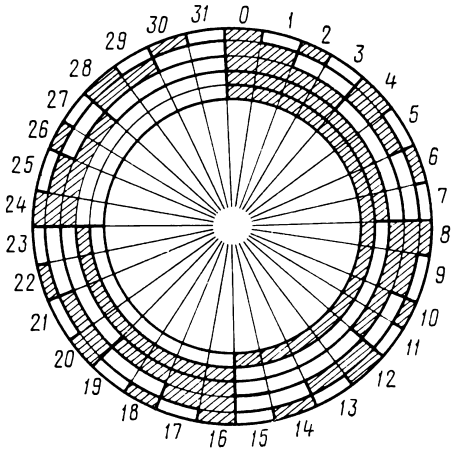

Fig. 8.4. Binary code wheel

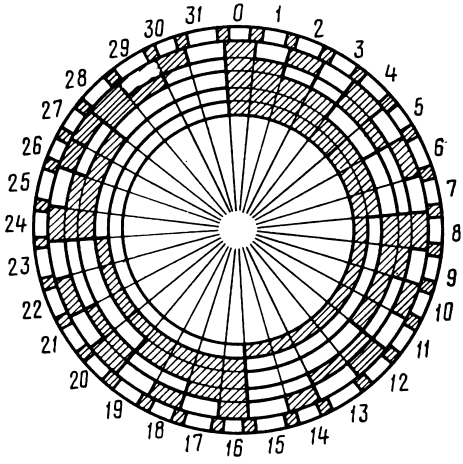

Fig. 8.5. Binary code wheel with a redundant track

on a reading azimuth, i.e., column-wise on a linear code mask and along a radius on a code wheel. As the mask or wheel moves past the reading elements, they sense 0's and 1's and generate appropriate output signals. As a result, a certain definite binary number is generated at the output for each position of the code mask or wheel.

Among the limitations of digitizers using a binary code mask or wheel are stringent requirements for the accuracy in the manufacture of a mask or wheel and in the positioning of the reading elements all on the same reading azimuth as already explained. Actually, there is always an error, however small, in both respects, so that the reading elements fail to cross the reading azimuth all at the same time. When crossing the reading azimuth

brings about the change of a digit in only one digit position (as may occur with binary coding) this miscoding error (or ambiguity) will not exceed the least significant bit. If, however, digits change in several positions as the reading elements cross the azimuth, the ambiguity error may well reach the maximum value of the quantity being digitized. For example, during the transition from the binary code group 01111 to the binary code group 10000, the digits must be swapped in all the positions. If this fails to happen, any five-bit numbers may occur at the azimuth, for example 00000 if the digits in the first bit positions change and that in the fifth position does not; or 00001 if the digits in the second, third and fourth positions change but those in the first and fifth remain as they were before. Similarly, other binary numbers may be generated, such as 11111 if the digit in the fifth position changes while those in the other positions do not.

To repeat, the ambiguity error in our example may be as great as the maximum value of the quantity. Because of this, binary code masks or wheels cannot be used in their unmodified form.

Several antiambiguity schemes exist. Consider some of them.

Limiting the reading zones. With this technique no code group is allowed to be sampled at the reading azimuth. This is done by adding one more track with zones marking the limits beyond which no code combination can be sensed (Fig. 8.5). With this redundant track, code groups can only be picked off within the light segments. The width of dark segments is decided by the tolerance on the manufacture of the code wheel or mask and on the positioning of the reading elements, and should span the total tolerance. When a reading element takes up a position within a dark segment on the redundant track, reading is inhibited by, say, disabling the reading elements on all other tracks (in all digit positions). With this technique, the limiting zones simulate lost motion, as it were — should the code wheel take up a position in which the reading elements are all within a limiting zone, no information will exist at the output.

To avoid this indeterminacy, the reading elements may be offset relative to the reading azimuth for the duration of the reading time, or the code wheel may be caused to dither while the reading elements are within a limiting zone.

Unfortunately, this technique greatly limits the capabilities of digitizers in terms of accuracy and speed. This is why their use may be warranted only where the accuracy of conversion is immaterial and the quantity being digitized varies at a slow rate.

"*Dual-brush*" *method.* With this technique, two reading elements (or brushes), A and B, are provided in each digit position, except the first, the two elements being offset from each other by half the unit in the particular digit position. At each reading instant, the element displaced from the reading azimuth is selected. This technique is embodied in a digitizer three stages of which are shown in Fig. 8.6.

The digitizer has three cams D_0, D_1 and D_2, each geared down to the next at a ratio of 1 : 2. Thus, conversion takes place within three digit positions of a binary number.

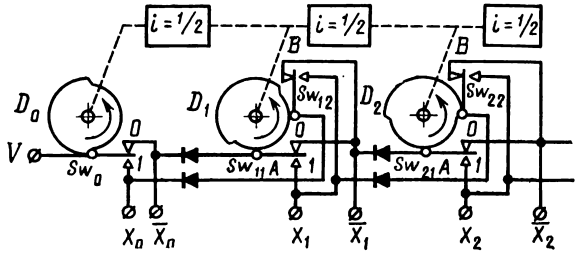

Fig. 8.6. A/D converter using mechanical switches

To avoid the ambiguity at the junction of code groups, each cam (except that in the least significant position) has two reading elements each in the form of a two-position switch (Sw_{11} and Sw_{12} for D_1, Sw_{21} and Sw_{22} for D_2). The switches in the same digit position are arranged to operate at times offset by half the quantum, that is, 90°. At each particular instant, reading will be done by the switch displaced from the reading azimuth of the respective cam. The switch is selected according to the position of the cam in the preceding digit position. To this end, the initial position of each cam is shifted relative to that of the cam in the preceding position by 1/4 of a quantum, or 45°.

According to the position of the switches, the code combinations are sampled at the terminals x_i or $\bar{x}_i$ in each digit position.

Where reading is done by photocells or induction pickoffs, preference is given to a logic selection of reading elements known as the *V-brush* or *Barker method.* Here, a straight binary code wheel or mask is used, but two reading elements, A and B, are provided in each digit position, except the first. The reading elements A are displaced through a quarter-width of the particular

digit position to the right from the reading element in the least significant position, and the reading elements B are displaced through a quarter-width of the particular digit position to the left (Fig. 8.7). Which reading element is used depends on what is read from the next least significant digit position, and it occurs at least 1/4 of a quantum from the reading azimuth.

A different approach can be taken to secure the offset between the reading elements and the reading azimuth (Fig. 8.8). The reading elements are arranged to lie on the same straight line. The code mask is constructed so that each digit position, except

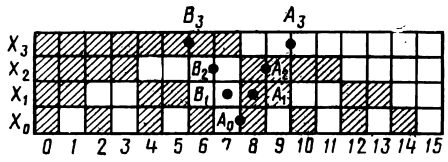

Fig. 8.7. Implementation of the V-brush method

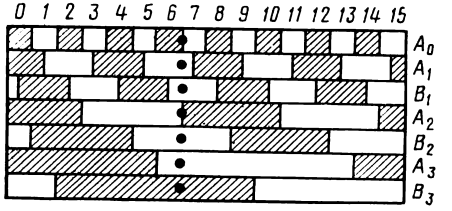

Fig. 8.8. Code mask based on the V-brush method

the first, has two sub-positions, A and B. The sub-positions A are displaced 1/4 of their width to the right from the junction of the first digit position, while the sub-positions B are displaced by the same amount to the left.

Which of the two techniques, shifted-brush or shifted-track, is preferable? This question can only be answered from a comparison of the two techniques in terms of circuit complexity and accuracy. In terms of accuracy, it pays to use a more elaborate code wheel or mask and simpler reading elements, but the wheel or mask will be greater in size.

The reading element used depends on what is read from the next least significant bit, and the choice is accomplished as follows. If a zero is read from the next least significant bit, the sub-position A will be read next. If a 1 is read, the sub-position B will be read next. To implement this logical sequence, use is made of a serial code, starting with the least significant bit.

The necessary switching can be done by a logical circuit such as shown in Fig. 8.9. It consists of a *NAND* gate, an *AND* gate, an *OR* gate, and a flip-flop, *FF*. Suppose that element A_0 which reads the least significant bit on the code wheel or mask feeds a 1 signal to input A of the *NAND* gate. Since the gate has not

been disabled by a previous inhibit signal, the 1 signal will pass on to the *OR* gate whence it will be written into the flip-flop, *FF*. Arrival of a reset pulse at the "*Reset*" input of the flip-flop will cause it to reset, and the 1 signal will pass on to output and, at the same time, to the inhibit input of the *NAND* gate and to the *AND* gate. At that instant, elements A_1 and B_1 in the next digit position ought to read their respective bits. However, the *NAND* gate will block the passage of any signal at that instant, while the *AND* gate will pass the bit picked off by element B_1, and this will be routed to the *OR* gate whence it will be written into the flip-flop. During the next clock time, another reset pulse will cause the flip-flop to reset, and the bit picked off by element B_1 will pass on to output and, at the same time, to the *NAND* gate and the *AND* gate. If, during a given clock time a 0 signal passes on to output, the *NAND* gate will not be disabled during the

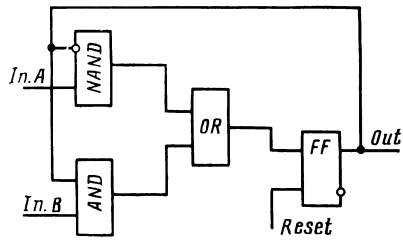

Fig. 8.9. Logic of the V-brush method

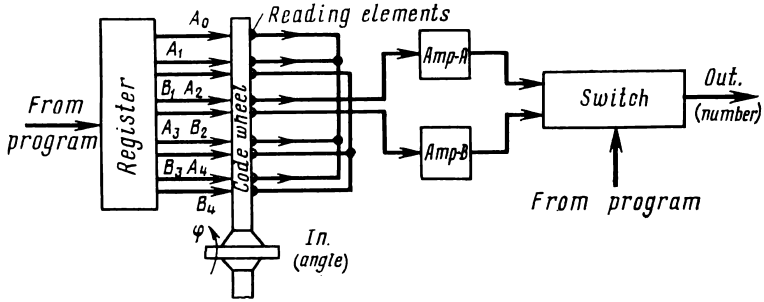

Fig. 8.10. Implementation of the V-brush method

next clock time and the code signal arriving at input *A* will be allowed to pass on to output. The reset pulses should reset the flip-flop in synchronism with the bit-by-bit reading of the code wheel (or mask), starting with the least significant bit.

The circuit of a digitizer using the V-brush method is shown in Fig. 8.10. It consists of a register which samples the reading elements in turn in response to the reset clock pulses applied to the flip-flop in the switching circuit, a code wheel, V-brushes,

amplifiers *Amp-A* and *Amp-B* to boost the signals coming from the brushes, and a switching circuit.

Among the advantages of the V-brush method is the fact that the offset of reading element A_1 relative to element B_1 increases with increase in bit significance. Owing to this, less stringent requirements need be satisfied as regards the accuracy of the code wheel (or mask) in the most significant positions and the positioning of the reading elements.

The use of the serial code extends the conversion time in comparison with reading in a parallel code. This limitation is avoided in the dual-brush method.

With the dual-brush method, as already noted, each digit position, except the first, has two reading elements, *A* and *B*. In contrast to the V-brush method, all reading elements *A* are displaced half the least significant bit to the left and arranged in-line, while all reading elements *B* are displaced one-half the least significant bit to the left and are likewise arranged in-line. Which reading elements are used, *A* or *B*, is determined by what is read from the least significant bit. If a 1 is read from the least significant bit (a light segment), the signal is taken from all elements *B*; if a 0 is read from the least significant bit, the signal is taken from all elements *A*.

Thus, while the V-brush method uses as many reading clock times as there are bits, the dual-brush method uses only two reading clock times. However, the same tolerance has to be specified on the positioning of the reading elements, whatever their bit position, so that the tolerance range is rather narrow in the case of multi-bit numbers.

Cyclic binary codes. As has been shown, ambiguous readings may result at the junction of two adjacent binary code groups when two or more bits are required to change at the same time. It has also been shown that this ambiguity, or miscoding error, may be kept to not over a unity in the least significant bit, if the change of a bit is allowed to occur in only one digit position. Hence it has been proposed to construct a code in which a transition from one binary code group to another would cause only one bit to change. Such a code uses binary sequences which possess such a property.

This type of code can be constructed, using the graphs of Fig. 8.11*a*, *b*. The intersections of columns and rows in these graphs produce four-bit code groups. The first two bits are defined by the row No., and the remaining ones, by the column No.

A condition for a code in which only one bit will change during a transition from one binary code group to another is that no move is allowed along the diagonal or a repeated move along any row of a column. In moving along the path shown by the

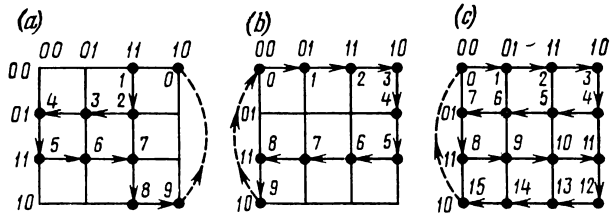

Fig. 8.11. Cyclic codes

heavy line in the graph, the code combination appearing at the nodal points (0 through 9) is defined by the intersections of the respective column and row. The code combinations for these cases are presented in Table 8.1.

Table 8.1

Decimal number	0	1	2	3	4	5	6	7	8	9
Graph *a*	0010	0011	0111	0101	0100	1100	1101	1111	1011	1010
Graph *b*	0000	0001	0011	0010	0110	1110	1111	1101	1100	1000

As is seen, the two graphs produce different code combinations for the same decimal numbers, but in each case no more than one bit changes at the junction of two adjacent code groups. The number of nodal points, columns and rows in a graph depends on the number of quantizing steps to be used.

In order that the code group associated with the last nodal point will differ from that of the first in only one bit, the graphs should be constructed in a cyclic manner; hence the codes synthesized in this way are called *cyclic*.

Instead of a cyclic graph, use may be made of a cube. With a great range of numbers, however, the procedure becomes prohibitively more complicated.

The choice of a particular path in which the code wheel or mask of a digitizer should be read and, as a consequence, the choice of a particular cyclic code depends on the ease of conversion from a binary to cyclic code and, especially, from the cyclic to binary code. Such a conversion is necessary because cyclic codes are inconvenient for arithmetic operations and are seldom used in digital computers.

A relatively simple conversion from a binary to cyclic code and back is offered by the *Gray*, or *reflected binary, code.* It is synthesized, using the graph of Fig. 8.11*c*. The decimal numbers from 0 through 15 and their equivalents in the Gray and straight binary codes are shown in Table 8.2.

Table 8.2

Decimal number	Normal binary	Reflected binary	Decimal number	Normal binary	Reflected binary
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

A code wheel and a code mask using the Gray code are shown in Fig. 8.12. As is seen, the white and black segments in all bit

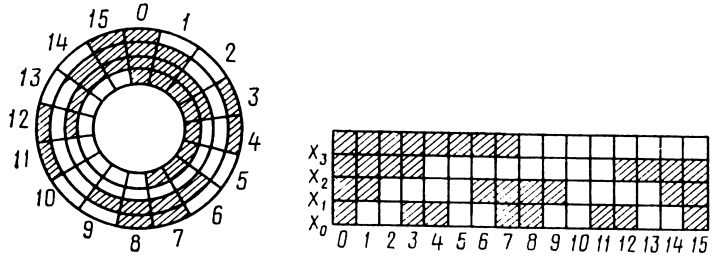

Fig. 8.12. Gray code wheel and mask

positions except the most significant one, are twice as wide as those used in normal binary coding. As a result, the accuracy of coding is doubled for the same resolution (the value of the least significant bit).

If the normal binary number is represented as $x_{n-1}x_{n-2} \ldots x_1x_0$ and the reflected (Gray) binary number as $\alpha_{n-1}\alpha_{n-2} \ldots \alpha_1\alpha_0$, then one set of rules for conversion is:

$$\begin{aligned}
\alpha_{n-1} &= x_{n-1} \\
\alpha_{n-2} &= x_{n-2} \oplus x_{n-1} \\
\alpha_i &= x_i \oplus x_{i+1} \\
&\cdots \\
\alpha_0 &= x_0 \oplus x_1
\end{aligned}$$

where x_i and α_i are a 0 or a 1 bit in the normal binary and Gray code, respectively.

For example, if the normal binary number $x_4x_3x_2x_1x_0$ is 11001, then using the above set of rules, its reflected binary equivalent, $\alpha_4\alpha_3\alpha_2\alpha_1\alpha_0$, will be

$$\begin{aligned}
\alpha_4 &= x_4 = 1 \\
\alpha_3 &= x_3 \oplus x_4 = 1 \oplus 1 = 0 \\
\alpha_2 &= x_2 \oplus x_3 = 0 \oplus 1 = 1 \\
\alpha_1 &= x_1 \oplus x_2 = 0 \oplus 0 = 0 \\
\alpha_0 &= x_0 \oplus x_1 = 1 \oplus 0 = 1
\end{aligned}$$

that is, 10101.

Thus, one needs a suitable logical circuit in order to convert normal binary numbers into their Gray code representation.

One set of rules to convert a Gray code number into its normal binary representation is:

$$\begin{aligned}
x_{n-1} &= \alpha_{n-1} \\
x_{n-2} &= x_{n-1} \oplus \alpha_{n-2} \\
&\cdots \\
x_i &= x_{i+1} \oplus \alpha_i
\end{aligned}$$

The logic circuit implementing the above set of rules is shown in Fig. 8.13. The conversion proceeds serially, starting with the most significant bit.

The number to be converted is fed bit-by-bit in Gray code representation to the *NAND1* gate and the *OR2* gate.

If the most significant bit is a 1, it will pass via the *OR3* gate to the *OR4* gate, whence it will go to the input of the flip-flop, *FF*. Upon arrival of a reset clock pulse, the number stored in the

flip-flop will be passed on to output and, in synchronism with the next bit in Gray representation, to the *OR1* gate and to the *NAND1* gate.

If the next bit is a 1, it will not be able to pass via the *NAND1* gate because the latter is disabled. Instead, it will pass to *NAND2* gate to disable it, too. As a consequence, the 1 signal will not be able to pass from the *OR1* gate to output, either. This is why, a 0 will be written in the next bit position of the output number. If, on the other hand, the next bit in the Gray number is 0 the *NAND2* gate will not be disabled, the 1 signal will pass from the *OR1* gate on to output, etc.

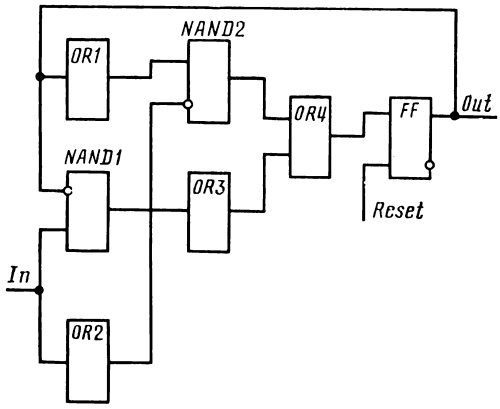

Fig. 8.13. Logic of Gray-to-straight binary code conversion, with the most significant bit first

Sequential digital computers utilize serial codes in which numbers are fed with their least significant bit first. Because of this, the method described above would necessitate the subsequent conversion of numbers from a serial code with its most significant bit first into a serial code with its least significant bit first. In such cases, it is more attractive to convert numbers in the Gray code into a serial code with the least significant bit first. This can be done by the set of rules explained below.

To begin with, one counts the ones in the Gray number to be converted. If it contains an even count of ones, a 0 is written in the least significant bit of the normal binary number; in the case of an odd count of ones, a 1 is written. The digits in the remaining bit positions are found by the following rules:

$$x_0 = \sum_i \alpha_i$$

$$x_1 = x_0 \oplus \alpha_0$$

$$x_2 = x_1 \oplus \alpha_1$$

$$x_{i+1} = x_i \oplus \alpha_i$$

For example, if the number in reflected binary code to be converted is 10101, then, by the above set of rules its normal binary form will be as follows:

$$x_0 = \alpha_0 \oplus \alpha_1 \oplus \alpha_2 \oplus \alpha_3 \oplus \alpha_4 = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 1$$
$$x_1 = x_0 \oplus \alpha_0 = 1 \oplus 1 = 0$$
$$x_2 = x_1 \oplus \alpha_1 = 0 \oplus 0 = 0$$
$$x_3 = x_2 \oplus \alpha_2 = 0 \oplus 1 = 1$$
$$x_4 = x_3 \oplus \alpha_3 = 1 \oplus 0 = 1$$

Thus, the normal binary number will be 11001. The logical circuit implementing these rules is shown in Fig. 8.14. It operates as follows.

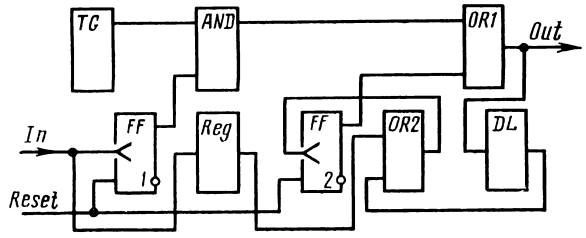

Fig. 8.14. Logic of Gray-to-straight binary code conversion, with the least significant bit first

The number in the Gray code is read, starting with its least significant bit, and the signal is applied to the complementing input of a toggle flip-flop, *FF1*. If there is an even count of ones, the flip-flop will reset on counting them, and the associated *AND* gate will be disabled. If there is an odd count of ones, the flip-flop will change state and enable the *AND* gate. The clock pulse arriving at the input of the *AND* gate from the timing generator, *TG*, passes on to the *OR1* gate, whence it passes on to output to represent a 1 in the least significant bit of the normal binary number. At the same time, the least significant bit of the normal binary number is routed via a delay line, *DL*, and another *OR* gate, *OR2*, to the complementing input of a second toggle flip-flop, *FF2*, which also accepts the least significant bit of the Gray number during the next clock time. Operating as a single-bit counter, *FF2* adds together the incoming bits and puts out the next bit of the normal binary number via *OR1* to output and, at the same time, into the delay line, etc.

In order to hold the Gray number while the least significa which i bit of the normal binary number is identified and also to feed the qua during appropriate clock times to *FF2*, there is a shift registe be conv *Reg*. The least significant bit of the Gray number arriving at th the follc input of this register will appear at its output after the leas significant bit of the normal binary number has appeared at th input to *FF2*. After a conversion cycle is completed, a reset puls resets the circuit in readiness for the next cycle.

Ternary code wheels and masks. A ternary code wheel or mas converts mechanical motion into a ternary number. This conver This sion is advantageous if the converted numbers will then be handl into r ed by computing elements using a ternary code.

To implement ternary coding by the same techniques as ar used with binary code wheels or masks, two binary wheels o masks are needed (Table 8.3). On the first wheel or mask, 2' are represented in the same way as 1's, that is, by white segment (Fig. 8.15*a*). The second code wheel only has segments to re present bit positions in which 2's occur, and these are made a white segments (Fig. 8.15*b*). Both wheels are made fast to th shaft whose angular motion is to be converted to a ternary num ber. A modulo 3 sum is then taken of the bits occurring withir the same bit position, except the last.

Table 8.

Decimal number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Ternary number	000	001	002	010	011	012	020	021	022	100	101	102	110	111	112	120	121	12
Code group on first wheel	000	001	001	010	011	011	010	011	011	100	101	101	110	111	111	110	111	11
Code group on second wheel	000	000	001	000	000	001	010	010	011	000	000	001	000	000	001	010	010	01

n a dig his can uantity he cosi nown a osine a puter of e more an conv he angu In con etween ortion o he origi ents diff

Function code wheels. The digitizers discussed so far conver wheel ha motion into a number which is a linear function of the motion f chang As often as not, however, one has to convert angular motio circuit c

which is a nonlinear function of some quantity. For example, if the quantity of interest is defined in polar coordinates and is to be converted to rectangular coordinates, one will have to solve the following equations:

$$z = \rho \sin \varepsilon$$

$$x = \rho \cos \varepsilon \cdot \cos \beta$$

$$y = \rho \cos \varepsilon \cdot \sin \beta$$

This can be done by first converting the primary data ρ, ε and β into normal binary numbers then finding the sines and cosines

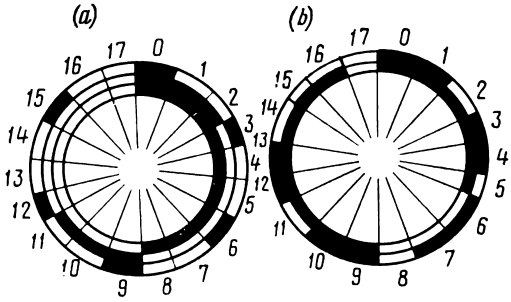

Fig. 8.15. Ternary code wheels

on a digital computer, and finally multiplying them together. Or this can be done in another way, that is, by converting an angular quantity or quantities directly into the digital representations of the cosine and sine. Code wheels (or masks) doing this are known as *function* code wheels (in our example, there will be a cosine and a sine code wheel). Such code wheels relieve the computer of some of the function conversions and may in some cases be more attractive than linear code wheels. Function code wheels can convert a broad class of continuous nonlinear functions of the angular position.

In constructing a function code wheel, balance has to be struck between the required and attainable resolution within the steepest portion of the function. For the sine function, this portion is near the origin of coordinates. Since the same quantizing step represents different angles, the black and white segments on the code wheel have to be made progressively longer as the slope (or rate of change) of the function decreases. Because of this, no logical circuit can be used to select the reading elements in order to

avoid ambiguity at the junction of two code groups. This is wh ection th cyclic binary codes have to be employed. ient is

A segment of a sine code wheel using the Gray code is show Operat in Fig. 8.16*a* and *b*. The sign is indicated by an additional trac he sens as is customary with digital computers, the "+" sign is repr on is t sented by a 0, and the "—" sign, by a 1. This sine code whe ucting s may be used as a cosine wheel, too, by providing additional rea ing elements displaced through 90° from the sine elements.

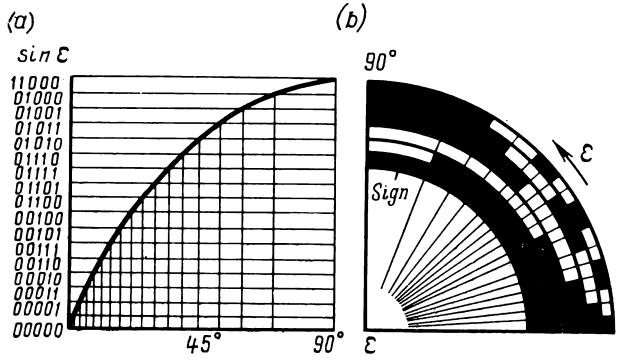

Fig. 8.16. Sine code wheel

Function code wheels are more difficult to make than linea code wheels. In a linear code wheel, the circumference is divide into segments of equal length; in a function code wheel this done according to the function to be represented. Prior to makir y a wir a function code wheel, it is necessary to digitize the function gr ther ma phically in terms of the angular position, and then to write th hould ha discrete values of the function in a cyclic binary code.

8.4. MATRIX CODING

here *m*

Converters utilizing matrix coding are not unlike increment ower of A/D converters extended to include a coding matrix. As will ff to the recalled, in an incremental A/D converter a brush or any oth In bina reading element moves along the code track, senses the states ie outpu the consecutive segments, and feeds appropriate signals to senses counter. It has already been noted that this form of conversio conve can only handle incremental changes in the quantity of interes hich pr In a matrix converter, the signals sensed by the brush are fed ower m a coding matrix rather than to a counter. For each segment resented

ection there is a separate line so that the signal from each segment is represented by a unique binary code combination.

Operation of a matrix converter will be clear from Fig. 8.17. he sensing brush is made fast to the shaft whose angular position is to be encoded. In operation, the brush rides current-conducting segments arranged in a circle. Each segment is connected

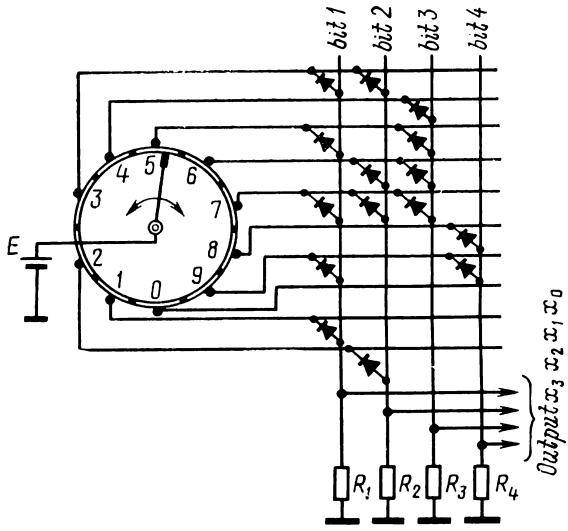

Fig. 8.17. Matrix coding

y a wire to a coding matrix. The latter may be a diode or any ther matrix encoder. For binary coding, this matrix encoder hould have n output wires:

$$n = \log_2 m$$

here m is the number of segments, usually a multiple of some ower of two (otherwise, the logarithm will have to be rounded ff to the next higher integer).

In binary coding, each wire is assigned to a particular bit in ie output binary number. As the brush moves round the circle, senses an appropriate voltage in each position, and this voltage conveyed over the respective wire to the matrix (encoder) hich produces the respective binary number on its output wires. ower may be supplied by a d.c. source if the numbers are resented by levels, or a pulsed power source if the numbers are

represented in a pulse-coded form. In the latter case, the supple, this
frequency should be chosen according to the rate of change of as would
quantity being converted, so that at least one cycle of change an alter
occur while the brush dwells at a segment. other by

Where the numbers are represented in pulse-coded form, a di energized
encoder which requires a great number of diodes for its reali
tion should preferably be replaced with a magnetic core ma
(such a matrix will have as many cores as there are bits in
output number). In binary coding, the wire from the first s
ment threads the first bit core (001), the wire from the sec
segment the second bit core (010), the wire from the third s
ment threads the first and second bit cores (011), etc. At readi
the signals appearing on the wires will represent the binary di
associated with the respective segments.

The sensing brush should be wider than the insulation betw
adjacent segments, otherwise the matrix would be de-energi
and no data would exist at the output when the brush is mov
between segments. However, a wide brush might overlap two s
ments and, as a result, two bits would be sensed at a time.
normal binary coding, this would lead to appreciable errors
milar to those arising at the junction of two binary code gro
on code wheels. As a proof, let the brush move from segment N
towards segment No. 4. The binary number appearing on
output wires is 0011, which is to change to the binary num
0100. At the instant when the brush touches both segments,
binary number actually appearing on the output wires will
0111, which differs markedly from the correct number.

As with code wheels, several anti-ambiguity schemes may
used, including cyclic binary codes. With them, only one bit atrix A/D
required to change in moving across the reading azimuth. U adapted
fortunately, this complicates the circuitry because the cyclic bin share all
code has then to be converted to normal binary.

Another anti-ambiguity scheme is to sample the segments d
ing two clock times. A normal binary code is used, but the wi
of even segments are connected to a pulsed power source duri typical in
a clock time t_1, and the wires of odd segments, during anot ediate co
clock time, t_2, displaced from the first by half the clock time. Psured by
of such a coding matrix using four cores is shown in Fig. 8 mulation
(cores *5* and *6* supply clock pulses). Since adjacent segments ersion, in
energized during different clock times, only one will be sampl is applic
even though the brush may touch both at the same time. As resolver c
result, a correct number is produced on the output wires. F).

ple, this will be either 0011 or 0100, and not the mutilated as would happen with simultaneous sampling.

an alternative, two brushes may be used, displaced from other by the width of insulation between the segments, and energized from a seperate power source.

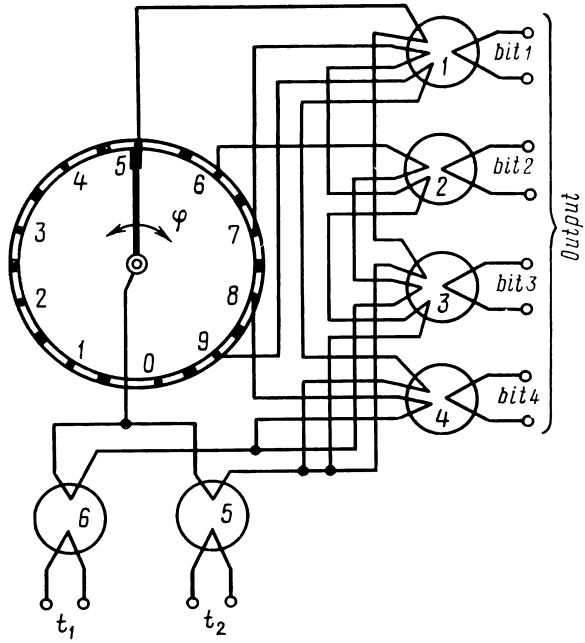

Fig. 8.18. Two-clock matrix coding

atrix A/D converters are fairly simple in design, but they are adapted to electric-contact (or brush) reading. Obviously, share all the drawbacks inherent in this reading technique.

8.5. INDIRECT CONVERSION

typical indirect analog-to-digital conversion involves an intediate conversion to time. The time interval can then be ured by counting the pulses needed to fill it. To avoid an mulation of errors, a complete new value is obtained at each ersion, independent of the previous one. This conversion me- is applicable to both shaft position (using magnetic-drum resolver converters) and electric quantities (by the ramp me-

Magnetic-drum A/D converter. This type of converter (Fig. 8 consists of a magnetic drum, *MD*, a write head, *WH*, read he *RH1* and *RH2*, an erase head, *EH*, and an electronic control cuit. The magnetic drum has two tracks; one is used by the sec read head, *RH2*, and the other, by all other heads. The write is made movable and is linked mechanically to the shaft whose gular position, φ, is to be encoded. The magnetic drum and write head are arranged on the same axis of rotation.

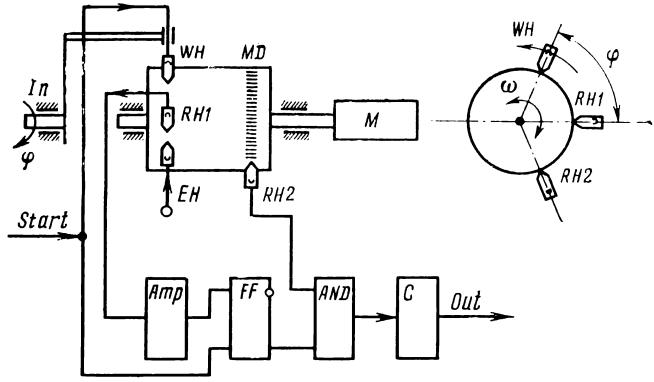

Fig. 8.19. Magnetic-drum A/D converter

The converter is started by a "*Start*" pulse from the con circuit. As a result, the movable write head writes a pulse m ing the angular position φ on the magnetic drum. At the s time, the "*Start*" pulse causes the flip-flop, *FF*, to change s and enable the *AND* gate. The timing pulses sensed by the sec read head from the magnetic drum are routed via the *AND* g on to the counter, *C*. Just as the marker pulse written by write head comes under the fixed read head, *RH1*, a pulse is nerated in the latter, which, on passing through an ampli *Amp*, resets the flip-flop, thereby terminating the counting that is, passage of timing pulses via the *AND* gate to the cou

Thus, an appropriate binary number is left in the counter portional to the rotation of the write head relative to the fi read head and, as a consequence, to the angular position, φ, be converted.

Before the control circuit applies another "*Start*" pulse, this ca counter must be cleared, and the marker pulse erased from drum by the erase head, *EH*.

Resolver te resolv examine converte **Induction** ergized, le propo at of the

ere $E =$
$M_1 =$
$M_2 =$
$i_1 =$
$i_2 =$

If

ere θ is en

This exp ar phase movabl d a c $= M\omega I$ constant ion, thi at the st nstant m nt, and e. Thes tisfied b ro) resol Consider taining this ca $= I_0 \sin \omega$ the oth

(Fig. 8**Resolver phase-shift A/D converter.** Before taking up a com-
read he:te resolver phase-shift A/D converter, it will be worth while
control examine what a resolver is and how an angular position can
y the sec converted to a time interval in the form of a phase angle.
e write H*nduction resolver.* If two coils at right angles to each other be
ft whose ergized, an emf will be induced in a third coil, with a magni-
um and le proportional to the angle between the axis of this coil and
at of the first two:

$$E = jM_1\, di_1/dt + jM_2\, di_2/dt$$

ıere E = emf in the third coil
M_1 = mutual inductance of the first and third coils
M_2 = same of the second and third coils
i_1 = current in the first coil
i_2 = current in the second coil

If

$$i_1 = I_0 \sin \omega t, \qquad M_1 = M \cos \theta$$
$$i_2 = I_0 \cos \omega t, \qquad M_2 = M \sin \theta$$

ıere θ is the angle between the axes of the first and third coils,
en

$$E = jM \cos \theta I_0 \omega \cos \omega t - jM \sin \theta I_0 \omega \sin \omega t$$
$$E = jM\omega I_0 \cos (\omega t + \theta)$$

the con This expression shows that the emf of the third coil has a li-
pulse ma ar phase shift proportional to the angle between the axes of
At the s movable and fixed coils,
hange sd a constant magnitude,
the sec $= M\omega I_0 =$ constant at $\omega =$
e *AND* g constant. For its implemen-
tten by ion, this method requires
pulse is at the supply currents be of
n ampli hstant magnitude, $I_0 =$ con-
nting cy nt, and in phase quadra-
the coun e. These requirements are
counter tisfied by induction (or syn-
to the fi ro) resolvers.
on, φ, be Consider another method for
taining a linear phase shift.
pulse, this case, one coil is energized with a sinusoidal current,
ed from $= I_0 \sin \omega t$ (Fig. 8.20), so that emf's of induction are induced
the other two at right angles to each other, respectively

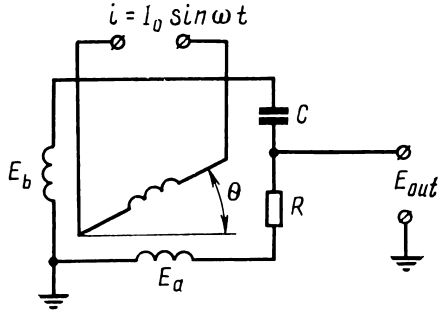

Fig. 8.20. Phase shift by splitting

defined as

$$E_a = jM_1\, di/dt$$

$$E_b = jM_2\, di/dt$$

Substituting $i = I_0 \sin \omega t$ gives

$$E_a = jM_1 I_0 \omega \cos \omega t$$

$$E_b = jM_2 I_0 \omega \cos \omega t$$

Applying the superposition principle (assuming that the circuit is linear) and neglecting the inductive reactance of the coils, the output voltage of each coil due to E_a and E_b will be

$$E_{a\ out} = \frac{jM_1 I_0 \omega R \cos \omega t}{R + 1/(j\omega C)}$$

On separating the imaginary and real parts, we get

$$E_{a\ out} = -\frac{M_1 I_0 \omega^2 RC \cos \omega t}{R^2\omega^2 C^2 + 1} + j\,\frac{M_1 I_0 \omega^2 R^2 C^2 \cos \omega t}{R^2\omega^2 C^2 + 1}$$

Hence, the magnitude and phase shift of output voltage are:

$$|E_{a\ out}| = \frac{M_1 I_0 \omega^2 RC}{R^2\omega^2 C^2 + 1}\sqrt{1 + \omega^2 R^2 C^2}$$

$$\varphi_1 = \tan^{-1}(-\omega RC)$$

Thus, the output voltage due to E_a may be defined as

$$E_{a\ out} = \frac{M_1 I_0 \omega^2 RC}{\sqrt{1 + R^2\omega^2 C^2}} \cos\left[\omega t + \tan^{-1}(-\omega RC)\right]$$

The output voltage due to E_b is

$$E_{b\ out} = \frac{jM_2 I_0 \omega \cos \omega t}{[R + 1/(j\omega C)]\, j\omega C}$$

Proceeding as before, the magnitude and phase shift of output voltage may be defined as:

$$E_{b\ out} = \frac{M_2 I_0 \omega^2 RC \cos \omega t}{1 + R^2\omega^2 C^2} + j\,\frac{M_2 I_0 \omega \cos \omega t}{R^2 C^2 \omega^2 + 1}$$

$$|E_{b\ out}| = \frac{M_2 I_0 \omega}{\sqrt{1 + R^2 C^2 \omega^2}}$$

$$\varphi_2 = \tan^{-1}(1/\omega RC)$$

Finally,

$$E_{b\ out} = \frac{M_2 I_0 \omega}{\sqrt{1 + \omega^2 R^2 C^2}} \cos\left[\omega t + \tan^{-1}(1/\omega RC)\right]$$

The total output voltage is the sum of $E_{a\,out}$ and $E_{b\,out}$, that is

$$E_{out} = E_{a\,out} + E_{b\,out} = \frac{I_0\omega}{\sqrt{1+\omega^2R^2C^2}}\{M_1\omega RC\cos[\omega t+\tan^{-1}(-\omega RC)] + M_2\cos[\omega t+\tan^{-1}(1/\omega RC)]\}$$

Noting that

$$M_1 = M\cos\theta$$
$$M_2 = M\sin\theta$$

and arranging ωRC to be unity by adjustment of R and C, we get

$$E_{out} = (MI_0\omega/\sqrt{2})[\cos\theta\cos(\omega t-45°)+\sin\theta\cos(\omega t+45°)]$$

Writing $\cos(\omega t-45°)$ as $\sin(\omega t+45°)$, we finally have

$$E_{out} = (MI_0\omega/\sqrt{2})\sin(\omega t-45°+\theta)$$

This expression shows that output voltage has a phase shift f $E_{out\,max} = MI_0\omega/\sqrt{2} =$ constant, which is achieved when $\omega =$ $=$ constant.

This method is simpler to implement than the previous one. This is because instead of maintaining the phase shift between current in two mutually perpendicular coils to a high degree of accuracy, we now only need to adjust the values of R and C so as to satisfy the equality $\omega RC = 1$. This method can be implemented with ordinary induction resolvers in which the rotor and stator have each two mutually perpendicular coils (Fig. 8.21).

Synchro. A linear phase shift can be obtained with synchros. A synchro has three stator windings, s_1, s_2 and s_3, spaced 120 electrical degrees apart and energized with a three-phase voltage (Fig. 8.22).

The emf of induction in the rotor winding is

$$E = jM_1\,di_1/dt + jM_2\,di_2/dt + jM_3\,di_3/dt$$

Recalling that

$$i_1 = I_0\sin\omega t \qquad M_1 = M\cos\theta$$
$$i_2 = I_0\sin(\omega t-120°) \qquad M_2 = M\cos(\theta-120°)$$
$$i_3 = I_0\sin(\omega t+120°) \qquad M_3 = M\cos(\theta+120°)$$

and substituting the above expressions in the previous equation, we get

$$E = jM \cos \theta I_0 \omega \cos \omega t + jM \cos (\theta - 120°) I_0 \omega \cos (\omega t - 120°) + jM \cos (\theta + 120°) I_0 \omega \cos (\omega t + 120°)$$

After trigonometric manipulation, we may finally write

$$E = (3/2) jMI_0 \omega \cos (\omega t - \theta)$$

The above equation shows that the rotor voltage of a synchro has a constant amplitude and a linear phase shift relative to

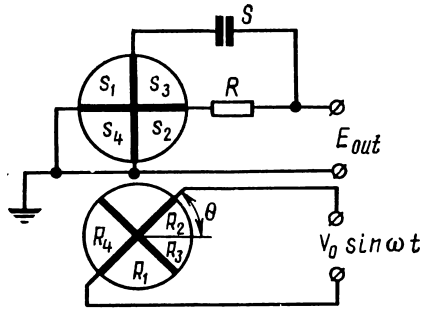

Fig. 8.21. Connection of an induction resolver in a circuit

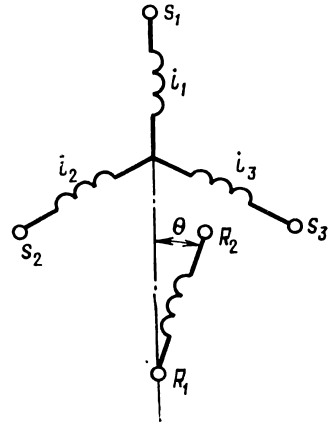

Fig. 8.22. Windings of a synchro

supply voltage. A major advantage of the resolvers examined above is that they are commercial items readily available on the market.

Among their disadvantages is a relatively low accuracy.

Inductosyn. An *Inductosyn* is a multipolar induction transducer which, like the synchro resolver examined above, generates electric signals which are sine and cosine functions of angular position. In contrast, however, an Inductosyn is more accurate in shaft position-to-digital conversion (within a second of arc), owing to a great number of poles on the rotor and stator and the fact that the coil conductors are deposited ("printed") on a glass substrate.

An Inductosyn consists essentially of two insulating discs. One is stationary and called the *stator*; the other is movable and

called the *rotor*. The two discs arranged on the same shaft are separated by an air gap. One or several windings are deposited on the mating surfaces of the discs; the windings may be made continuous or sectionalized. The windings look like radial strips alternately connected together. There should be an even number of strip conductors, so that the magnetic fluxes of the conductors will be opposite in sense everywhere.

The rotor and stator windings of an Inductosyn are arranged according to the end use of the device. For example, they may be

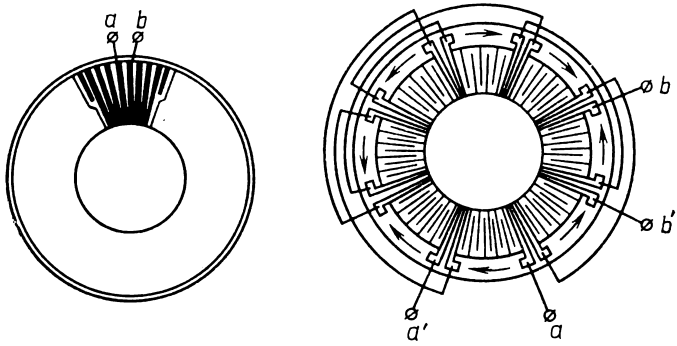

Fig. 8.23. Rotor and stator of an Inductosyn

identical or different. Say, the stator may have two multipolar sections, and the rotor, only one multipolar winding (Fig. 8.23). This is the most commonly used form of the Inductosyn. The stator windings are displaced from one another by half the pole pitch of the rotor. The rotor and stator windings may differ in winding pitch, but the ratio of the conductor width, l_0 (in angular units) to the conductor spacing, $l - l_0$, must be constant:

$$l/(l - l_0) = \text{constant}$$

where l is the angular distance between the centres of adjacent conductors, or the conductor pitch.

Twice the conductor pitch is called the *winding pitch*, τ:

$$\tau = 2l$$

When the stator winding is energized with a sinusoidal voltage of constant frequency, the emf induced in the rotor winding has an amplitude and a phase which are functions of the angular position of the rotor. Using an appropriate supply voltage for the

stator winding, it is possible to keep the amplitude constant while causing the phase to vary as a function of the angular position of the rotor. Conversely, it is possible, while keeping the phase constant, to cause the amplitude to vary as a function of the angular position of the rotor.

The phase of output voltage is maintained constant by feeding one stator winding with a sinusoidal voltage, and the other with a cosinusoidal voltage, while both have the same phase. Then the voltage taken from the rotor winding will vary sinusoidally in amplitude. The Inductosyn is then said to operate in the *amplitude* mode.

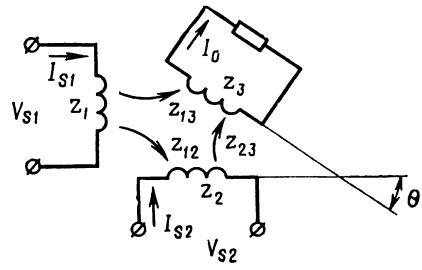

Fig. 8.24. Connection of windings in an Inductosyn

If the stator windings are energized with voltages of the same amplitude but shifted in 90° in phase, the voltage taken from the rotor will be constant in amplitude but its phase will be varying as a function of the angular position. The Inductosyn is then said to operate in the *phase* mode. As phase can be converted to a number more accurately than amplitude, the phase mode is the basic form of operation in cases where an Inductosyn is used as a shaft position-to-number converter. We shall examine this mode of operation in greater detail with reference to a rotary Inductosyn such as shown in Fig. 8.23. Connections between the windings of the Inductosyn are shown in the diagram of Fig. 8.24.

When the stator windings are energized with voltages

$$V_{s1} = V_s \sin \omega t$$

and

$$V_{s2} = V_s \cos \omega t$$

the voltage across the rotor winding will be

$$\dot{V}_r(\varphi) = \dot{K}_v V_{s1} \cos \varphi + \dot{K}_v V_{s2} \sin \varphi \qquad (8.1)$$

where $\dot{K}_v = \dot{V}_r / V_s$ is the complex transfer function of the transducer and φ is the phase parameter connected to the angular position of the rotor as

$$\varphi = \theta n = \theta (360/\tau) \qquad (8.2)$$

where n is the number of winding pitches or periods having the same phase, and τ is the winding pitch.

It is seen from Eq. (8.2) that the phase parameter of the Inductosyn varies n times faster than the angular position of the rotor, and increases with decreasing winding pitch, τ. In other words, the change in angular position being converted is scaled up.

Now that we have solved the problem of converting shaft position to phase shift, the next step is to measure this phase shift. Owing to scaling-up, this can be done to a higher level of accuracy.

For the case in question, the electric quantities are connected by the following set of equations

$$\left.\begin{aligned} z_1\dot{I}_{s1} + z_{12}\dot{I}_{s2} + z_{13}\dot{I}_r &= \dot{V}_{s1} \\ z_{21}\dot{I}_{s1} + z_2\dot{I}_{s2} + z_{23}\dot{I}_r &= \dot{V}_{s2} \\ z_{31}\dot{I}_{s1} + z_{32}\dot{I}_{s2} + z_3\dot{I}_r &= 0 \end{aligned}\right\} \tag{8.3}$$

where $\dot{I}_{s1}$ and $\dot{I}_{s2}$ = current in the first and second stator windings, respectively

$\dot{I}_r$ = current in the rotor winding

z_1, z_2 = impedance of the first and second stator windings, respectively

z_3 = total impedance of the rotor winding and load

z_{12}, z_{21} = mutual impedance of the first and second stator windings

z_{13}, z_{31} = mutual impedance of the first stator winding and rotor winding

z_{23}, z_{32} = mutual impedance of the second stator winding and rotor winding

Solving the set of equations, Eqs (8.3) for the current in the rotor winding, we get:

$$\dot{I}_r = \frac{\dot{V}_{s1}(z_{21}z_{32} - z_2z_{31}) - \dot{V}_{s2}(z_1z_{32} - z_{12}z_{31})}{z_1z_2z_3 + z_{12}z_{23}z_{31} + z_{13}z_{21}z_{32} - z_1z_{23}z_{32} - z_{12}z_{21}z_3 - z_{13}z_2z_{31}} \tag{8.4}$$

To avoid instrumental errors in the Inductosyn, it is important to observe the following conditions:

(1) the impedance of the stator windings should be the same in magnitude:

$$z_1 = z_2 = z \tag{8.5}$$

(2) the stator windings should be at right angles to each other, so that their mutual impedances cancel out:

$$z_{12} = z_{21} = 0 \tag{8.6}$$

(3) the mutual impedances of the stator and rotor windings must vary sinusoidally (cosinusoidally) with the angular position of the rotor:

$$\left.\begin{aligned} z_{13} = z_{31} &= -j\omega M_{13} \sin n\theta \\ z_{23} = z_{32} &= +j\omega M_{23} \cos n\theta \\ M_{31} &= M_{23} = M \end{aligned}\right\} \tag{8.7}$$

where M_{13} and M_{23} are the maximum mutual inductances of the respective stator and rotor windings;

(4) the supply voltages of the stator windings should be of the same amplitude:

$$V_{s1} = V_{s2} = V_s \tag{8.8}$$

(5) in the phase mode of operation, the supply voltages should be as follows:

$$\left.\begin{aligned} \dot{V}_{s1} &= V_{s1} \sin \omega t \\ \dot{V}_{s2} &= jV_{s2} \sin \omega t \end{aligned}\right\} \tag{8.9}$$

Noting Eqs (8.5) through (8.9), the expression for current, Eq. (8.4), after substitution takes the form

$$\dot{I}_r = \frac{V_s \sin \omega t \cdot zj\omega M \sin n\theta - jV_s zj\omega M \cos n\theta}{z_2 z_3 - z\,[j\omega M (\cos n\theta)^2] - z\,(-j\omega M \sin n\theta)^2}$$

After simple manipulation, we finally get

$$\dot{I}_r = \frac{\omega M V_s \sin \omega t}{zz_3 + \omega^2 M^2} (\cos n\theta + j \sin n\theta) \tag{8.10}$$

As is seen, the magnitude of current in the rotor winding is a function of the angular position of the rotor and its phase varies n times faster than the angular position of the rotor:

$$\arg \dot{I}_r = n\theta$$

This relation holds for an ideal case. Actually, it is violated because the amplitudes and phases of the stator winding fields are anything but equal, the stator windings are coupled, the rotor emf contains higher harmonics, the stator and rotor windings are not perfectly concentric, the discs are out of parallelism, and so on.

Through proper attention to these inaccuracies and a well-conceived design, it is possible to build Inductosyns with an accuracy to within a few seconds of arc.

Capacitive resolvers. In a capacitive resolver, mechanical motion is converted to an electric quantity by a capacitance transducer. The capacitance C between its plates is given by

$$C = k\varepsilon S/d = \varepsilon S/(0.36\pi d) = 0.885\varepsilon S/d \qquad (8.11)$$

where ε = permittivity
S = area of overlap between two plates, cm^2
d = plate separation
C = capacitance
k = proportionality factor

Equation (8.11) shows that the capacitance is determined by the plate separation d and the area of overlap S.

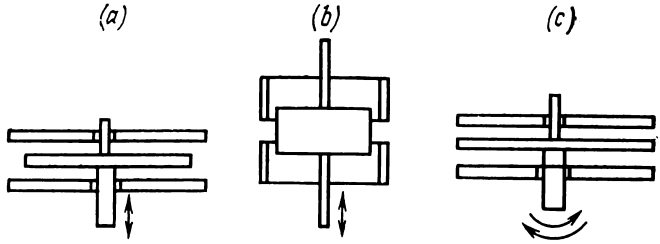

Fig. 8.25. Capacitance transducers

Physically, a capacitance transducer may have either a variable separation d and a fixed overlap area S, or a variable overlap area and a fixed plate separation.

A fixed-gap capacitance transducer usually consists of three parallel plates arranged parallel to one another. The outer plates are made fixed and form the stator, while the middle plate is free to move to and fro between the fixed plates and is called the rotor (Fig. 8.25*a*).

Variable-area, fixed-gap capacitance transducers have proved more convenient and found a wider field of application. They come in several designs: with the rotor free to move to and fro (Fig. 8.25*b*), with a rotor revolving on its shaft (Fig. 8.25*c*), with a movable pickoff, and with a fixed pickoff.

Where the stator has two plates, the transducer is said to be *two-lobe*; a *three-lobe* transducer has a stator made up of three plates. The rotor of a three-lobe capacitance transducer is shaped in a special way.

To enhance the accuracy of conversion, what may be called electric scale-up is utilized in capacitance transducer. Scale-up may be achieved through the use of a greater number of plates in the stator and a greater number of sinewave cycles in the rotor, or only in the number of sinewave cycles (for a three-lobe transducer, the number of sinewave cycles ought not to be a multiple of three, since the rotor would become symmetrical with respect to the stator and the output signal would be zero).

As a rule, the second method is used to obtain a 2-to-1 electric scale-up, because any further increase in the number of sinewave cycles in the rotor would bring down the amplitude of the output signal. The number of plates in the stator is chosen such that its ratio to that of sinewave cycles in the rotor be a multiple of 3/2 or 3. The stator plates are combined into groups. For proper synchronism, it is essential that for the maximum angular position of the rotor, θ_{max}, the electric angle does not exceed 2π.

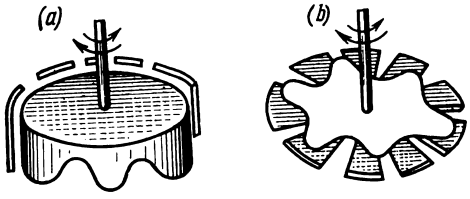

Fig. 8.26. Capacitance transducer (*a*) cylindrical; (*b*) flat

From this condition, the electric scale-up factor is defined as

$$n = 2\pi/\theta_{max} \tag{8.12}$$

For example, if it is specified that $\theta_{max} = 30°$, then $n = 6$. In such a case, the rotor will accommodate six sinewave cycles, and the stator will have nine plates assembled into three groups of three plates each.

Physically, a three-lobe capacitance transducer may be cylindrical or flat. In a cylindrical transducer (Fig. 8.26*a*), the stator is a hollow cylinder from an insulating material on the inside surface of which are attached metal plates, with every third plate interconnected.

In a flat capacitance transducer (Fig. 8.26*b*), the stator plates are arranged to lie all in the same plane. Their number and interconnection are the same as in a cylindrical transducer. The rotor is an odd-shaped plate arranged concentrically and parallel with the stator. The rotor is shaped to cause its capacitance to vary in a predetermined fashion.

In a shaft-to-digital converter, a capacitance transducer operates as a capacitive phase-shift resolver. Connection of a capa-

citive resolver in a shaft-to-digital converter is shown in Fig. 8.27*a*.

For proper operation of a capacitance transducer as a phase-shift resolver it is essential that its capacitance obey the following law:

$$\left.\begin{aligned} C_1 &= C_0 + C_m \sin\varphi \\ C_2 &= C_0 + C_m \sin(\varphi + \pi/2) \\ C_3 &= C_0 + C_m \sin(\varphi + \pi) \\ C_4 &= C_0 + C_m \sin(\varphi + 3\pi/2) \\ \varphi &= n\theta \end{aligned}\right\} \tag{8.13}$$

The voltages to be applied to stator plates *1*, *2*, *3* and *4* should be as follows:

$$\left.\begin{aligned} e_1 &= E \sin\omega t \\ e_2 &= E \sin(\omega t + \pi/2) \\ e_3 &= E \sin(\omega t + \pi) \\ e_4 &= E \sin(\omega t + 3\pi/2) \end{aligned}\right\} \tag{8.14}$$

An equivalent circuit of a capacitance resolver is shown in Fig. 8.27*b*. The following equations may be written for this circuit:

$$\left.\begin{aligned} e_1 &= R_L(i_1 + i_2 + i_3 + i_4) + i_1(1/j\omega C_1) \\ e_2 &= R_L(i_1 + i_2 + i_3 + i_4) + i_2(1/j\omega C_2) \\ e_3 &= R_L(i_1 + i_2 + i_3 + i_4) + i_3(1/j\omega C_3) \\ e_4 &= R_L(i_1 + i_2 + i_3 + i_4) + i_4(1/j\omega C_4) \\ V_L &= R_L(i_1 + i_2 + i_3 + i_4) \\ e_1 &= -e_3 \\ e_2 &= -e_4 \end{aligned}\right\} \tag{8.15}$$

From Eqs. (8.15), the load voltage V_L is found to be

$$V_L = \frac{(e_1C_1 + e_2C_2 + e_3C_3 + e_4C_4)\,R_L j\omega}{(C_1 + C_2 + C_3 + C_4)\,R_L j\omega + 1} \tag{8.16}$$

Substituting Eqs. (8.13) and (8.14) into Eq. (8.16) and carrying out some manipulation, we get

$$V_L = \frac{2EC_mR_L\omega}{\sqrt{1 + 16C_0^2R^2\omega^2}} \sin(\omega t + n\theta + \pi/2) \tag{8.17}$$

It is seen that the phase of the output (load) voltage is a linear function of the angular position and changes at a rate n times faster. That is, electric "gearing-up" takes place, enabling the accuracy of reading to be improved.

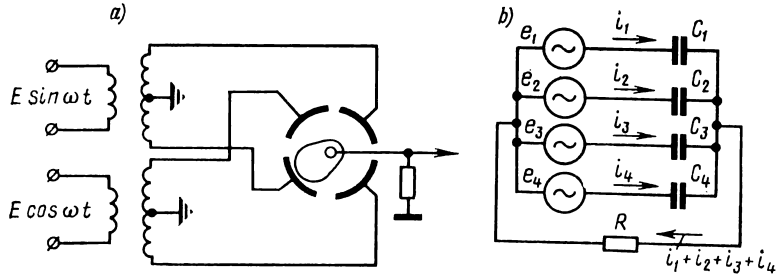

Fig. 8.27. Capacitance transducer
(*a*) connection in a circuit; (*b*) equivalent circuit diagram

A three-phase resolver may be made in the form of a three-lobe capacitance transducer. Then, the following voltages should be applied to plates *1*, *2* and *3*:

$$\left.\begin{aligned} e_1 &= E\sin\omega t \\ e_2 &= E\sin(\omega t + 2\pi/3) \\ e_3 &= E\sin(\omega t + 4\pi/3) \end{aligned}\right\} \tag{8.18}$$

Accordingly, the capacitances should vary as follows:

$$\left.\begin{aligned} C_1 &= C_0 + C_m\sin\varphi \\ C_2 &= C_0 + C_m\sin(\varphi + 2\pi/3) \\ C_3 &= C_0 + C_m\sin(\varphi + 4\pi/3) \\ \varphi &= n\theta \end{aligned}\right\} \tag{8.19}$$

In this case,

$$V_L = \frac{1.5EC_mR_L\omega}{\sqrt{1 + 9C_0^2R_L^2\omega^2}}\sin(\omega t + \psi - n\theta) \tag{8.20}$$

that is, as in the previous case, the phase of output voltage is a linear function of the angular position of the rotor, scaled up n times.

The electric scale-up factor n may run into several tens. However, an excessive increase in n would cut down the range of angular positions due to the relation $n = 2\pi/\theta_{max}$. This is why this type of resolver can in the general case be only used to convert small angular displacements or to give fine readings in multispeed converters.

Among the merits of capacitance transducers are small power consumption and simplicity of manufacture. Owing to electric scale-up, they ensure high precision in conversion.

Among their demerits is high internal resistance running into tens or even hundreds of megohms.

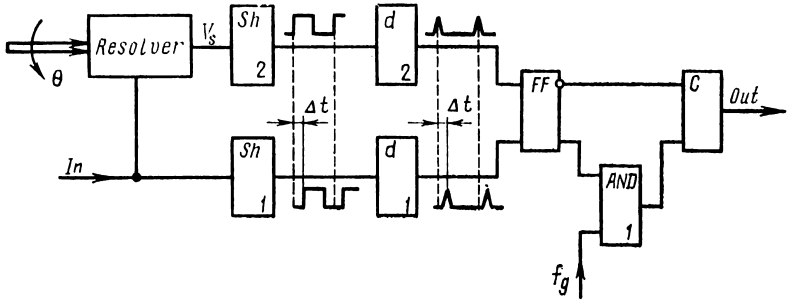

Fig. 8.28. Resolver phase-shift converter

As an example of a resolver phase-shift used as an A/D converter, we shall examine the simplified circuit of Fig. 8.28. The resolver generates a voltage E which is shifted in phase relative to the supply voltage V_s by an amount proportional to the angular position of the rotor. These voltages are then shaped by shapers, *Sh*, differentiated and limited by differentiating networks, *d*, to form two sets of unipolar pulses. One set due to the supply voltage V_s is independent of the angular position of the rotor, while the time shift Δt of the other relative to the first is a linear function of the angular position. The two sets of pulses alternately drive a flip-flop which in turn enables and disables an *AND* gate. When the *AND* gate is open, pulses f_g supplied by a pulse generator pass through it to an up-counter, *C*. The number N of these pulses received by the counter depends on their frequency f_g and time interval Δt:

$$N = f_g \Delta t$$

The time interval may be expressed in terms of the angular position of the rotor, θ, and the frequency of the magnetic field, f_{mf}, as

$$\Delta t = \theta°/360 f_{mf}$$

Then the number of pulses stored by the counter over a cycle (depending on the angular position) will be

$$N = (f_g/360 f_{mf})\,\theta°$$

As is seen, the precision of the circuit depends on the frequency stability of the pulse generator supplying timing pulses to the counter, and the stability of the supply frequency of the induction resolver. The errors due to these sources may be rectified by locking the two frequencies to a time standard or by locking one frequency to the other so that their ratio f_g/f_{mf} is constant, $f_g/f_{mf} = \text{constant}$.

The precision with which shaft position is converted to digital form is mainly affected by the error in phase shift due to the rotation of the moving coils in the induction resolver. Letting this error be close to 1°, that is, 1/360 of a circle, the conversion can be accomplished with an accuracy of 1/256 of a circle, so that an eight-digit counter will do.

Where a higher precision is important, a two-speed system has to be used, that is, one giving a fine and a coarse reading

The repetition frequency f_g of the pulses fed to the counter is chosen such that the counter will be filled full in one complete revolution of the magnetic field:

$$f_{mf} = f_g/2^n$$

where n is the number of digits in the up-counter.

8.6. MULTISPEED A/D CONVERTERS

It often happens that the desired precision of motion-to-digital conversion is more than can physically be obtained with the reading method adopted. For example, the precision of the electric-contact (brush) method does not exceed 9 or 10 digit positions, that of an inductive pickoff not over 8 to 9 digit positions, that of synchro resolvers 8 or 9 digit positions, etc. As a way out, resort is made to what are called *multispeed converters* or coders. In them, a code wheel is coupled to a second by a reduction gear

train, the second to a third, etc. In fact, each is a *convertor* having a certain precision.

Suppose the desired precision in terms of bits is n and the precision attainable with the reading method adopted is m, such that $n > m$. Then, these m bits are arranged on one wheel, and the remaining $n - m$ bits on a second.

The wheel reading the least significant bits is called a *fine converter*, and the wheel reading the most significant bits is called a *coarse converter*. For simplicity and standardization, both wheels (coarse and fine) are made with the same number of bit

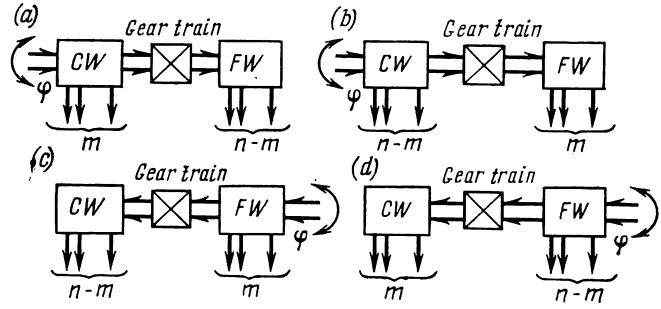

Fig. 8.29. Two-speed converters

positions, using all of them, m, on the first, and the remainder, $n - m$, on the second. If $n - m = m$, all bit positions will be utilized on the second wheel, too. If, on the other hand, $n - m \neq m$, which is usually the case, either $n - m$ most significant bits or $n - m$ least significant bits may be utilized for implementation.

As already noted, the code wheels are coupled to one another by a reduction gear train. The gear ratio in this train is chosen according to the manner in which the bit positions are arranged on the code wheels, which of the wheels is coupled to the shaft whose angular position is to be converted, that is, which of the code wheels is the input one. Generally, four distinct cases may arise (Fig. 8.29).

Case 1 (Fig. 8.29*a*). The input shaft is that of the coarse code wheel, *CW*. All of the m bit positions physically implementable on the coarse wheel are utilized. On the fine code wheel, *FW*, use is made of $n - m$ least significant bit positions. The gear ratio in this case must be $1 : 1/2^{n-m}$, that is, an up-gear train will

be used. If $n - m$ most significant bit positions be used on the fine code wheel, the gear ratio will be $1 : 1/2^m$.

Case 2 (Fig. 8.29*b*). The input shaft is that of the coarse code wheel. On this wheel, $n - m$ most significant bit positions are used, and all the m bit positions on the fine code wheel. The gear ratio of the gear train is $1 : 1/2^{n-m}$, that is, an up-gear train will be used. If $n - m$ least significant bit positions be used on the coarse code wheel, the gear ratio will be $1 : 1/2^m$.

Case 3 (Fig. 8.29*c*). The input shaft is that of the fine code wheel, *FW*. All the m bit positions of the fine code wheel are utilized, and the remaining $n - m$ bit positions, on the coarse code wheel, *CW*, using its most significant bits. The gear ratio is $1 : 2^{n-m}$, that is, a down-gear must be used. If the least significant bit positions be used for the remaining $n - m$ bits, the gear ratio will be $1 : 2^m$.

Case 4 (Fig. 8.29*d*). The input shaft is that of the fine code wheel. On this wheel, $n - m$ least significant bit positions are utilized, and the remainder is arranged on the coarse code wheel, *CW*. The gear ratio in this case is $1 : 2^{n-m}$, that is, a down-gear train must be used. If $n - m$ most significant bit positions be used on the fine code wheel, the gear ratio will be $1 : 2^m$.

As is seen, the gear ratio may be $1 : 1/2^{n-m}$ and $1 : 2^{n-m}$ or $1 : 1/2^m$ and $1 : 2^m$ according as the $n - m$ bits arranged on the fine or coarse wheel and according as the fine or coarse wheel is used as input.

To simplify the matters and to reduce load on the input shaft, preference should be given to a down-gear train with a lower gear ratio. The use of up-gear trains which put on an increased load, especially dynamic, is limited.

An appreciable error is caused by the backlash of the reduction gear train. The backlash for each of the spur, screw and bevel gears is

$$\Delta = kA/(mz)$$

where Δ = backlash of a gear pair in units of the transmitted quantity
m = pitch of the gear
z = number of teeth in a gear
A = speed factor of the shaft carrying the gear
k = coefficient of precision (determined by tolerances on manufacture and assembly), type and design of gear

train. Its values for unadjustable and adjustable shafts are listed in Table 8.4

Table 8.4

Accuracy class	Unadjustable shafts				Adjustable shafts
	k for large-gear diameter of				
	under 30 mm	30-100 mm	100-200 mm	over 200 mm	
2	0.023	0.028	0.035	0.040	0.009
3	0.036	0.042	0.053	0.058	0.025

The overall backlash, Δ_Σ, for the entire reduction gear train is the sum of the backlashes of each pair, that is

$$\Delta_\Sigma = \sum_{i=1}^{n} \Delta_i$$

The overall backlash of a reduction gear train depends on the accuracy class, pitch and number of gear teeth and also (in terms of speed factor) on the gear ratio of each pair and of the reduction train as a whole.

The speed factor A_2 of a driven shaft is connected to that of a drive shaft, A_1, by the gear ratio, i, as follows:

$$A_2 = A_1/i$$

The reduction gear trains of multispeed converters are expected to meet stringent requirements for accuracy because the instrumental errors of the gear train would be directly absorbed in those of the converter as a whole. The tolerances, especially in the most significant bit positions, may be relaxed through the use of the V-brush method.

The use of a reduction gear train in a multispeed A/D converter enhances the precision of conversion in comparison with single-speed converters, but the instrumental errors of the gear train limit the overall digit range to 12 or 13 places.

The fine and coarse code wheels of a multispeed converter may be coupled by means other than a reduction gear train. Figure 8.30 shows a two-speed coder in which the wheels are coupled optically.

The shaft, *1*, of the coder carries a coarse code wheel, *2*, and a transparent disc, *5*, on which opaque segments are arranged round the circumference. Coaxially with, and close to, the transparent disc is a similar but stationary disc, *4*, on which the number of opaque segments is increased by one. On one side, the discs are illuminated by a ring light source, *3*. On the other side of the discs are two diametrically opposite light detectors, *6* and *7*, mounted on a shaft, *8*. As disc *5* is rotated, a pattern is observed on the light-detector side, in which extended opaque segments appear because the movable and fixed discs have different numbers of opaque segments. As the movable disc turns through one division, these extended segments rotate through 360°. Thus, the angular displacement is scaled up by a factor equal to the number *i* of divisions on the movable disc.

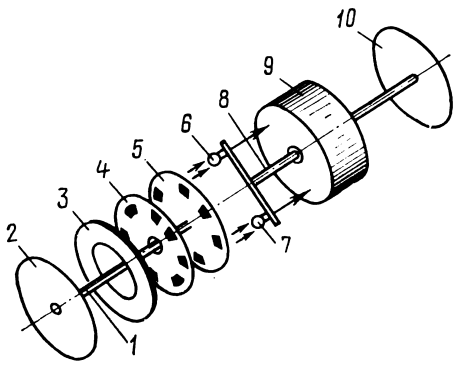

Fig. 8.30. Optical multiplicator converter

Rotation of extended opaque segments is sensed by light detectors *6* and *7* and followed by a servo mechanism, *9*, so that the light detectors and shaft *8* rotate in synchronism with the opaque segments, that is, *i* times faster. Made fast to shaft *8* is a fine code wheel, *10*. If the coarse code wheel, *2*, accommodates *m* digit positions, then the fine code wheel can hold the complementing $(n - m)$ bit positions, through a suitable choice of the number *i*.

This arrangement is free from inertia load inherent in a mechanical reduction gear train. However, the precision limitations of mechanical gear trains remain in force (the precision with which the extended opaque segments can be followed).

So far we have been dealing with multispeed converters used to extend the digit range (the number of significant places). It has been assumed that the angular position being converted ranges from 0 to 360°. Another application of multispeed converters may be the conversion of shaft position varying through several revolutions with the same digit range as furnished by single-speed coders. In this case, the fine code wheel converts shaft position within a single revolution, and the coarse code

wheel counts whole revolutions. If the total number of revolutions allowed is n, the coarse code wheel will have $m = \log_2 n$ digit positions, and the gear ratio of the train coupling the two wheels must be $1:2^m$. As before, the reduction gear train is expected to satisfy stringent requirements for accuracy and, as before, it is a source of undesirable additional loading.

The use of a reduction gear train in a multispeed converter may be avoided, if a separate coarse code wheel is replaced by a spiral mask made directly on the fine code wheel.

8.7. READING METHODS

Whatever the conversion method used (incremental, total-value, or indirect via time interval), A/D converters may use several reading techniques, such as electric-contact (or brush), photoelectric, electromagnetic, etc.

Electric-contact (brush) reading. This is the simplest and most widely used method. The signals sensed by the brushes are strong enough to make further amplification unnecessary. The brushes may be of the cam-driven and the sliding-contact type. Cam-driven brushes are used in conjunction with a lobed code wheel. This arrangement is shown in Fig. 8.31. The code wheel has on its circumference high and low spots representing binary 1's and binary 0's, respectively. A combination of a single brush and a single lobed code wheel gives an incremental converter. When the code wheel is rotating, the brush, B, completes the circuit each time it encounters a high spot, and a pulse representing a binary 1 appears in the output circuit. The number of output pulses counted over the conversion time gives the change in the angular position φ over that time.

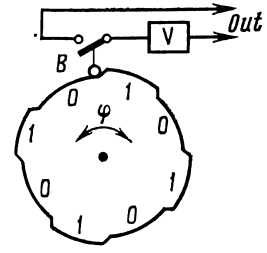

Fig. 8.31. Cam-driven brush

For direct reading, the converter should have as many code wheels as there are bits in the output binary number (and each code wheel should represent a certain definite bit position). The angular dimension of the low and high spots on each wheel should be in proportion to the significance (or weight) of a 1 and 0 in the particular digit position. The combination of high and low spots and their relative arrangement on each wheel should represent the code adopted. In the case of the V-brush

method, each bit, except the first, should have two wheels and two brushes, or one wheel but two brushes displaced from each other as explained earlier. The wheels should be made fast to each other and to the shaft whose angular position is to be converted. High and low coding spots may be made in the form of concentric tracks on the face of a wheel, but such a wheel is much more difficult to make. On the other hand, the cam-driven brushes examined above are well known in engineering practice and no difficulty arises in their manufacture. Among their demerits are considerable friction torque and a limited rate of conversion because of contact chatter occurring at high rates.

Sliding-contact brushes are used in conjunction with code wheels having conducting and nonconducting segments. While in a converter with cam-driven brushes, the contacts make and break under the action of brushes which are in turn actuated by the code wheel, in a sliding-contact converter, each brush has a contact tip which rides the surface of the code wheel. Obviously, the contact process is more complicated.

The reliability of contact in this arrangement depends on the quality of the insulating and conducting materials used in the code wheel and brush. The contacts should have a low contact resistance and high mechanical strength. To satisfy these requirements, the contacts are often made of noble metals or their alloys. The best choice is offered by alloys of the platinum metals which, apart from platinum, include ruthenium, rhodium, palladium, osmium and iridium. Pure platinum is too soft to be made into contacts, but it becomes sufficiently hard when alloyed with other metals. Ruthenium and osmium are hard metals, but they are not used in pure state either, because they are volatile when hot. Rhodium is very hard and resists corrosion well; because of this it is used to coat the conducting segments on code wheels. Sometimes, use is made of its substitute, the metal palladium which is less expensive, but has a lower hardness in comparison with rhodium.

Because the brush wears more than any point on the code wheel, it should preferably be made of a material which is harder than the conducting segments on the wheel.

The magnitude of contact resistance is a function of contact pressure — the greater the pressure, the lower the contact resistance. Unfortunately, high contact pressure increases friction torque and speeds up wear. This is why, some sort of trade-off is inevitable in converter design.

Photoelectric reading. This technique owes its popularity to the high resolution it gives. For its operation, it utilizes light. More specifically, a transparent or an opaque segment on a code wheel is assigned to each quantum of the quantity being converted. On one side of the code wheel is a light source, and on the other, a light detector (a photocell). The light detector does or does not generate an output pulse according as a transparent or an opaque segment intercepts the light beam on its way from the source to the detector. As before, an output pulse may represent a binary 1, and no pulse, a binary 0.

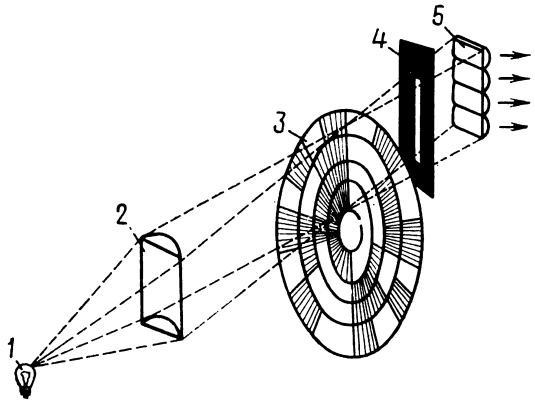

Fig. 8.32. Photoelectric train

To sum up, the photoelectric train of a converter comprises (Fig. 8.32) a light source, *1*, an optical train, *2*, to shape the light beam, a code wheel, *3*, with alternating transparent and opaque segments, a reading slit, *4*, to limit the light beam, and light detectors, *5*. All the elements of the photoelectric section are matched for proper operation.

The resolution of the code wheel, which is decided by the value of the least significant bit, must be matched to the width and intensity of the reading beam so that the electric signal generated by the light detector can unambiguously identify transparent and opaque segments. Obviously, the transition from a transparent to an opaque segment, or vice versa, must be indicated with the same accuracy as the segments themselves are made. However, this can only be achieved with an infinitesimal beam width and with the beam landing on the code wheel at precisely right angles. Unfortunately, a narrow beam would reduce the

luminous flux reaching the detectors, and these would register it with reduced certainty. On the other hand, a wider beam would carry more luminous energy, but the change in the output signal of the light detector at the boundaries of segments (at the reading azimuth) would become more gradual, and it would be more difficult to note the instant when a transparent segment changes to an opaque one.

The foregoing is illustrated in Fig. 8.33 for a varying beam

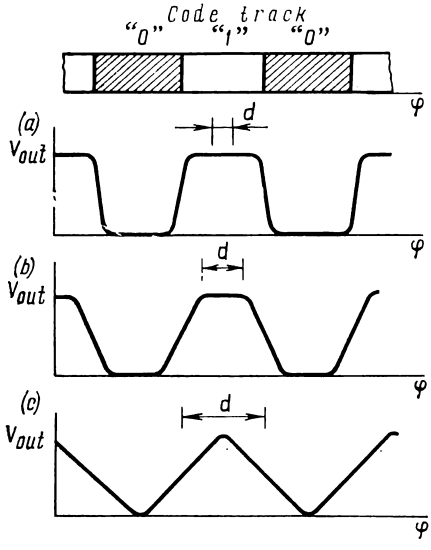

Fig. 8.33. Output signal as a function of beam width

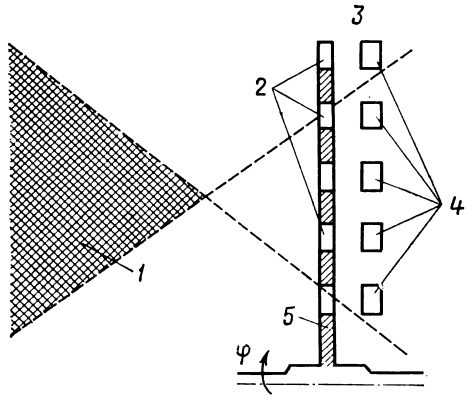

Fig. 8.34. Cross-talk

width, d, and a constant value, $\Delta\varphi$, of the least significant bit on the code wheel. The highest accuracy in discerning a binary 0 and a binary 1 on the code wheel is obtained with the narrowest reading beam (Fig. 8.33*a*). In this case, the output signal, V_{out}, has the greatest slope (rate of change). As the beam width, d, increases (Fig. 8.33*b* and *c*), the slope of the output signal decreases. The effective beam width is limited by the width of the segment representing the 1 in the least significant bit position (Fig. 8.33*c*). Any further increase in beam width reduces the amplitude of the output signal representing a 1. As a result, the difference between a "1" and a "0" decreases until they become undistinguishable.

The location and characteristics of the light sources and light detectors should be chosen so as to avoid cross-talk and to secure the highest 1-to-0 signal ratio.

Cross-talk may arise when light incident on one code track reaches the light detector of another; as a result, a false signal may appear at the output of that light detector. Cross-talk can be avoided through a proper location of the light source relative to the code wheel, *3*, and through a proper choice of width for the concentric opaque tracks, *5*, between transparent code tracks, *2* (Fig. 8.34). Cross-talk is inevitable when the light source is placed outside the shaded area, *1*. A point light source may be placed at any point within this area, while for a linear source the optimal area decreases.

Another form of cross-talk occurs when light passing through several transparent segments within the same track strikes its light detector. This drawback can be avoided through the use of slits to limit light to a narrow radial area only.

The relative position of the light source, code wheel, reading slit and light detectors must be such that no cross-talk can occur, and enough radiant flux can reach the light detectors. Also, proper match must be secured between the light source and detectors.

Consider the light detectors and light sources used in photoelectric converters.

LIGHT DETECTORS. These are various photocells. A *photocell* is a semiconductor device placed in an evacuated or a gas-filled envelope, whose electric properties are functions of the incident luminous flux. At present, A/D converters mostly use photovoltaic cells and partly photoresistors. Although they all convert light to electricity, these devices widely differ in the processes that take place inside them and cannot therefore be described in terms of the same parameters and relations.

Photovoltaic cells. These are devices which depend for their operation on the photoconductive effect that takes place at a $P - N$ junction. A device with a single $P - N$ junction is called a *photodiode*, and that with two or three $P - N$ junctions, a *phototransistor.*

As is the case with other electronic devices, the operation of a photovoltaic cell is described by its volt-ampere characteristic. The volt-ampere characteristic of a silicon photovoltaic cell is shown in Fig. 8.35 where V_s is the source voltage. For practical purposes, it can be described by an expression of the form

$$I_{ph} = I_d + V/R_L$$

where I_{ph} = primary photocurrent, that is, the total current of the photovoltaic cell due to the action of incident

radiation (primary photocurrent is practically independent of temperature)
V = voltage across the $P - N$ junction
R_L = load resistance
I_d = dark current leaking through the $P - N$ junction

$$V/R_L = I_L$$

$$T_d = I_s \exp(qV/kAT) - 1$$

where I_s = saturation current, that is, the value to which the dark current tends as $V \to -\infty$ (saturation current greatly increases with rising temperature)
$q = 1.6 \times 10^{-19}$
C = electronic charge (in coulombs)
V = voltage across the photovoltaic cell
$k = 1.4 \times 10^{-23}$ J/deg = Boltzmann's constant
A = dimensionless constant determined by the cell material (for germanium photodiodes, $A = 1$)
T = temperature, K

Figure 8.36 shows an equivalent circuit of the photovoltaic cell which answers the above equations. As is seen, the load does not remain constant; the resistance of the $P - N$ junction traversed by current changes with the photocurrent I_{ph}.

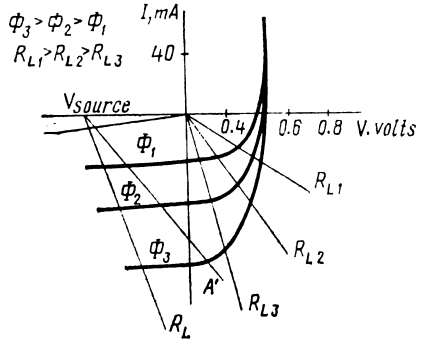

Fig. 8.35. Volt-ampere characteristics of a silicon photocell

The equations quoted above do not take account of the resistance, R_c, between the barrier layer and contacts, and the resistance, R_m, of the barrier layer, shown by dotted lines in the equivalent circuit diagram. These resistances affect the photocurrent, but the change is insignificant in many cases.

According to the polarity in which voltage is applied to the $P - N$ junction, a photovoltaic cell may operate either in the photovoltaic mode (Fig. 8.37*a*) or in the diode mode (Fig. 8.37*b*).

In the photovoltaic mode, the current and voltage take different signs (the fourth quadrant in Fig. 8.35) and, as a result, the cell has a negative static resistance. In other words, the cell acts as a current generator and needs no external source to sustain its

operation. In this mode, there is no dark current and internal noise is low.

In the diode mode, both the current and the voltage take the same "—" sign (the third quadrant in Fig. 8.35), and the cell has a positive static resistance. Now the cell needs a supply of external voltage for its operation, and this voltage should be applied so as to reverse-bias the cell. A major advantage of the diode mode is the generation of high-signal voltages, because the

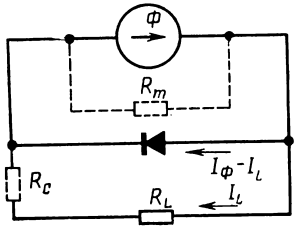

Fig. 8.36. Equivalent circuit diagram of a photocell

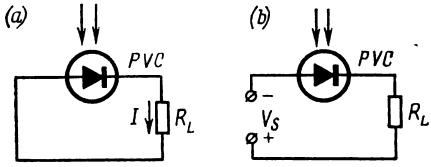

Fig. 8.37. Operating modes of a photovoltaic cell

cell may be connected to a high-resistance load. Unfortunately, noise rises appreciably and increases with rising voltage and stray light.

In the photovoltaic mode, the load resistor is traversed by a photocurrent which is part of the primary current the other part of which is the leakage current through the $P - N$ junction.

In the diode mode, apart from the photocurrent the load resistor carries the dark current flowing in the same direction. Thus, the direction of flow of dark current changes according as the cell is forward or reversed biased.

The sensitivity of photovoltaic cells drops markedly at high voltages and stray illumination.

The dependence of photocurrent I_{ph} on the incident luminous flux is called the *light characteristic of a photovoltaic cell.* This characteristic is nonlinear in the photovoltaic mode, and linear over a wide range of luminous flux Φ in the diode mode. Also, the photocurrent is affected by the spectral composition of the incident luminous flux.

The ratio of the current in the short-circuited external (load) circuit ($R_L = 0$) to the luminous flux in the case of a linear light characteristic is *integral luminous sensitivity*, symbolized

as K and expressed in mA/lumen or μA/lumen:

$$K = I_{ph}/\Phi$$

When the light characteristic is nonlinear, this ratio is termed the *mean integral luminous sensitivity* over the selected range of luminous flux values.

In the case of monochromatic radiation, Φ_λ, that is, radiation at a particular frequency or wavelength, a photovoltaic cell has what is called *spectral luminous sensitivity* defined as

$$K_\lambda = I_{ph}/\Phi_\lambda$$

The spectral luminous sensitivity of a cell varies with the wavelength of incident radiation. The spectral luminous sensitivity of

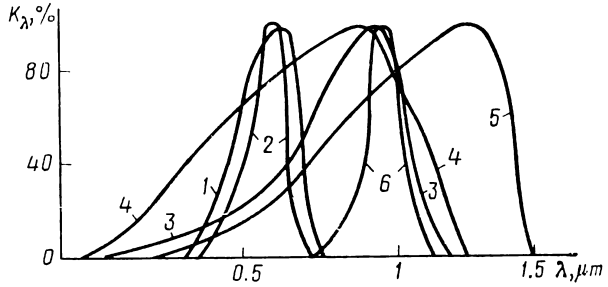

Fig. 8.38. Spectral sensitivity characteristics of photovoltaic cells
1—selenium without a corrective coating; *2*—selenium with a corrective coating; *3*—thallium sulphite; *4*—silver sulphite; *5*—germanium; *6*—silicon

a cell plotted against the wavelength of incident radiation yields its *spectral sensitivity curve.* The relative spectral sensitivity characteristics ($K_\lambda/K_{\lambda max} = K_\lambda$ in per cent) for some photovoltaic cells are shown in the plot of Fig. 8.38. As is seen, each curve shows a distinct peak, that is, each particular cell shows the greatest sensitivity for radiation at a particular wavelength. Outside such a spectral region, the sensitivity rolls off rapidly, which stresses the importance of matching the spectral characteristics of the light source and detector.

The basic specifications of some photovoltaic cells are summarized in Table 8.5.

In operation, the sensitivity of a photovoltaic cell falls off fairly rapidly at first (the cell is seasoned or stabilized) and at a relatively slow rate afterwards (as the cell ages, while in germanium photodiodes the saturation current rises and sensi-

Table 8.5

Photovoltaic cell	Max. integral luminous sensitivity, μA/lumen	Wavelength at maximum spectral luminous sensitivity, μm	Max. value of spectral luminous sensitivity, mA/W
Selenium	600	0.59	350
Silver-sulphide	8000	0.8-0.9	400
Thallium sulphate	11,000	0.95	500
Germanium	30,000	1.4-1.5	1000
Silicon	7000	0.95	850

tivity decreases). After a photovoltaic cell is de-energized, it will usually recover its performance.

Soviet-made photovoltaic cells are marked as follows: selenium cells, К (К-5, К-10, К-20); silver-sulphide cells, ФЭСС (ФЭСС-У-5, ФЭСС-У-3, ФЭСС-У-8, ФЭСС-У-10); germanium cells, ФД (ФД-1, ФД-2, ФД-3); silicon cells, ФДК (ФДК-1, ФДК-1в).

Selenium photodiodes have integral luminous sensitivity in the range 250-500 μA/lumen; thallium-sulphide cells, 5000-10,000 μA/lumen; silver-sulphide cells, 3500-8000 μA/lumen; germanium photodiodes, 10,000-20,000 μA/lumen; and silicon photodiodes, about 3000 μA/lumen.

When operated in the diode mode, silicon photodiodes show low operating voltage and dark current. The latter usually is below 1 μA and rises as high as 5 to 10 μA at a temperature of +60° C. Germanium photodiodes have a higher dark current which is 10 to 20 μA under normal conditions.

Apart from photodiodes, as already noted, use is made of phototransistors with a $P-N-P$ junction. Phototransistors have a higher sensitivity and their dark current is less temperature-dependent than that of photodiodes. In other respects phototransistors and photodiodes are identical in properties. Type ФТ-1 phototransistors have an integral luminous sensitivity of 170 to 500 μA/lumen, and type ФТГ-2 phototransistors, 2000 to 7000 μA/lumen.

The manner in which a phototransistor is connected in a circuit is shown in Fig. 8.39. Its volt-ampere characteristic (for a type ФТГ-1) is not unlike that of a junction transistor connected in a common-emitter circuit.

In circuit calculation, the load current for a specified radiation is found by plotting the load line and extending it until it intersects the volt-ampere characteristic of the cell at the specified value of illumination.

In the diode mode, the load line is described by an equation of the form

$$V = V_s - I_L R_L$$

where V_s is the supply voltage, I_L is the load current, and R_L is the load resistance.

In the photovoltaic mode, $V_s = 0$ and the load-line equation takes the form

$$V = -I_L R_L$$

If the volt-ampere characteristic of a photovoltaic cell is not known in advance, one has then to use the original state (equilibrium) equation and solve it simultaneously with the load-line equation, which poses appreciable difficulties.

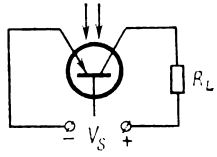

Fig. 8.39. Connection of a phototransistor

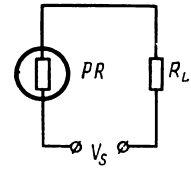

Fig. 8.40. Connection of a photoresistor

Photoresistors. These are semiconductor light-sensitive devices the electric resistance of which decreases with increasing illumination due to the photoconductive (inner photoelectric) effect. Connection of a photoresistor in a circuit is shown in Fig. 8.40. Photoresistors may be energized from both d.c. and a.c. sources.

Operation of a photoresistor can be described by an equation of the form

$$I_{ph} = AV^{\gamma}E^{\alpha}$$

where A = constant dependent on the material and design of the photoresistor
γ = nonlinearity factor of the volt-ampere characteristic
α = nonlinearity factor of the light characteristic

Since the volt-ampere characteristic is assumed to be linear, the sensitivity, like the photocurrent, is proportional to voltage. This is why data sheets usually quote relative sensitivity, that is sensitivity per volt, rather than sensitivity.

Photoresistors show high sensitivity over a relatively wide range of frequencies from the ultraviolet to the near infrared, and a stable resistance. With heating, their dark current increases.

Soviet makers are currently producing photoresistors from lead sulphide (type ФСА), cadmium sulphide (type ФСК and СФ2), and cadmium selenide (types ФСД and СФ3).

LIGHT SOURCES. Basically, the light sources used in photoelectric A/D converters are to meet the following requirements:

(1) the spectrum of the luminous flux should be matched to the spectral characteristic of the photodetector;

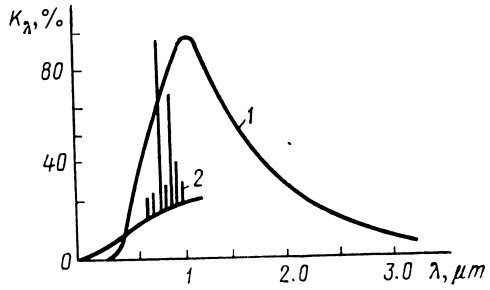

Fig. 8.41. Spectral response of lamps
1—tungsten-filament lamp; *2*—zirconium lamp

(2) the light beam should be concentrated to a small diameter, which requires that the luminous volume of the light source be as small as practicable (point-like or linear);

(3) the light beam should carry much power at high stability;

(4) service life should be as long as possible;

(5) power supply requirements should be kept to a minimum.

Light sources for photoelectric A/D converters may be continuously emitting lamps and flash tubes. The former are mostly incandescent lamps; the latter are various gas-discharge tubes.

In an incandescent CW lamp, light is given up by a filament fabricated from some refractory metal or metallic compound. Of pure metals, the best choice is tungsten which is ordinarily used for this purpose.

Incandescent lamps may be evacuated and gas-filled. Gas-filled lamps suffer from appreciable thermal losses. Tungsten lamps, both vacuum and gas-filled, show a peak emission in the short infrared region (Fig. 8.41). For example, a vacuum lamp using a tungsten filament at a temperature of 2500 K shows an emission peak at $\lambda = 1.5$ μm, and a gas-filled lamp at a filament temperature of 3000 K, at $\lambda = 0.96$ μm. The emission spectrum of

such lamps is a good match to the spectral characteristics of the light detectors discussed above.

The temperature of the filament is the key parameter which governs all other characteristics of a lamp.

Another important characteristic of a lamp is its *luminous efficiency*. It is defined as the ratio of the total luminous flux emitted to the total power consumed. The more the luminous flux a lamp emits for the power consumed, the more efficient it is.

Projector lamps suit photoelectric A/D converters best, because almost all of their luminous flux can be directed from the filament into the optical train. Projector lamps using a helical filament arranged at right angles to the lamp axis produce a light beam with a negligible angle of spread in a vertical plane and a considerable spread in a horizontal plane.

As a rule, incandescent lamps last for not more than 100 h. Lamps used in car headlights operating on 11 V and 23 V have a service life of 30 h, and their luminous flux is 6000 to 7000 lumens.

Apart from a filament, CW lamps may utilize an arc discharge. An example is offered by a zirconium lamp where an arc discharge takes place in zirconium vapours to which an amount of argon has been added. Outwardly, a zirconium lamp looks like an ordinary small gas-discharge tube. A distinction of this lamp is that it emits both a continuous and a line spectrum. The continuous spectrum comes from the small white-hot cathode spot which is uniformly luminous and has sharply defined edges. The line spectrum originates in the excited gas (argon) and zirconium vapours. Because the discharge is concentrated within the small area of the cathode spot, a very bright light beam is produced. Electrons are supplied by the thin layer of zirconium metal which emits at about 3000 K. Zirconium lamps operate on d.c. power. The working voltage is 130 V, but the starting voltage is very high, being from 1000 to 2000 V. This is a major limitation of zirconium lamps. A major advantage is that a zirconium lamp is turned on instantaneously, which fact enables the light beam to be modulated (chopped) by turning power on and off.

Another example of the continuously emitting gas-discharge lamp is the cesium-vapour lamp which produces infrared radiation of high intensity (Fig. 8.42). These lamps are available in 50-W, 100-W and 500-W sizes. A 100-W lamp is a tube 125 mm long and 35 mm in diameter, filled with cesium vapour and some inert gas (for example, argon). The arc in the lamp strikes

between two helical tungsten electrodes coated with barium and strontium oxides. The lamp shows an emission peak at 0.86 μm and 0.89 μm. A very valuable property of cesium-vapour lamps is that current can be modulated to a considerable depth.

Flash tubes. These tubes produce high-intensity, short-duration flashes of light. These flashes may be single and multiple and utilize various physical phenomena. Flash tubes for A/D converters are built to have a long service life and produce a multiplicity of flashes at a high rate. These requirements are satisfied by the strobotron, a glow lamp which produces flashes of light at a rate up to several kilohertz.

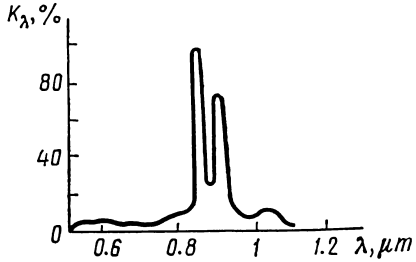

Fig. 8.42. Spectral response of a caesium lamp

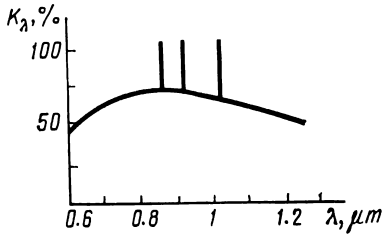

Fig. 8.43. Spectral response of a flash tube

A typical flash tube is a refractory-glass or quartz cylinder with an inside diameter of 1.5 to 10 mm. At the ends, the tube has sealed-in cylindrical nickel electrodes. The tube is filled with an inert gas (usually xenon, argon or krypton) at a low pressure.

The tube is fired by ionizing the filling gas. The resultant flash is very short (not over 10^{-6} s) and extremely bright. The duration of a flash depends on the ionization time of the gas and the impedance of the supply circuit. High-PRF tubes are filled with a mixture of argon and hydrogen. At a flash repetition rate of 1000 per second, this type of tube becomes conducting continuously, and it may be utilized only by connecting it in series with a separate switching tube which de-ionizes rapidly so that the capacitor can be re-charged as rapidly in time for the next flash.

As is seen from the plot of Fig. 8.43, flash tubes have a fairly wide emission spectrum. The emission peak occurs somewhere between 0.7 and 1.2 μm, which can conveniently be matched to the spectral response of light detectors.

In most cases, the firing is timed by means of a third electrode located inside the tube, to which a high-voltage firing pulse is applied. If a tube has no firing electrode, the firing can be timed by instantaneously raising the potential difference between the main electrodes. This can be done either by raising the voltage of the main power source or by building up a voltage pulse across a coil placed between the source and tube.

The power source of a flash tube should be capable of supplying a heavy current in a short pulse. Most often, the source is a capacitor which is charged from a relatively low-power d.c. supply and produces a strong pulse at discharge. Sometimes, to simplify the circuitry, a flash tube may be energized from a high-power a.c. source. Then the tube will fire at the instants when the voltage is a maximum and go out (or extinguish) when the voltage passes through zero. Flashes will occur at the frequency of the supply line.

From a comparison of continuous-emitting and flash tubes, we may conclude the following.

Continuous-emitting light sources are not critical with respect to power sources (which may be of low power rating). The continuous light beam does not limit the rate of change of the quantity being converted — no matter how often the light beam is interrupted by opaque segments on the code wheel or disc, this will be adequately sensed by a light detector of appropriate spectral response.

Major limitations of continuous-emitting light source are the relatively low intensity of light, heavy thermal losses which cause heating of other elements in the apparatus, appreciable time lag associated with the radiators because of which the light beam cannot be modulated through the power source and this has to be done elsewhere, usually past the light detector.

Among the merits of flash tubes are the high intensity of the light they emit and the pulsed working which makes any subsequent modulation (light-chopping) unnecessary.

Among the demerits of flash tubes are a rather elaborate circuitry of the power source and the need for a high firing voltage. Since flashes can only be produced at some finite rate, there is a limit to the rate at which the quantity being converted may change. Also, the luminous flux from flash tubes diminishes with time because the particles torn away from the cathode by bombardment deposit on the tube walls.

Induction reading. This technique is based on the use of a pulse transformer with an air gap in which a metal code wheel moves (Fig. 8.44). A slot in the wheel represents a binary 1 and a tooth a binary 0. The pulse transformer has a primary winding, W_1, and a secondary winding, W_2. During conversion, the primary is energized with interrogating current pulses. These pulses produce in the magnetic circuit a magnetic flux, Φ, which links the core of the primary winding, the air gap and the core of the secondary. As a result, an emf is induced in the secondary. When a slot enters the air gap, the amplitude V_1 of the emf in the secondary represents a binary 1. When a tooth enters the air gap, a markedly lower emf is induced in the secondary, with an amplitude, V_2, representing a binary 0. The reduction in the induced emf is due to the eddy currents existing in the code wheel, which give rise to a magnetic flux opposing the main flux. Because of this, the resultant magnetic flux linking the secondary is brought down by a considerable amount.

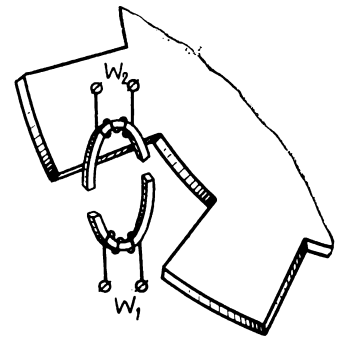

Fig. 8.44. Induction reading

As is seen, the primary and secondary cores and the code wheel form a single magnetic system, and they should be designed so as to take account of conflicting requirements. The leading ones may be stated as follows:

(1) the secondary emf should have the highest attainable amplitude, V_1, when reading a 1 (to meet this requirement, the air gap should be made as small as practicable, which in turn requires that the code wheel be extremely thin);

(2) the secondary emf should have the lowest attainable amplitude, V_2, when reading a 0 (to meet this requirement, the code wheel should be made from a metal producing considerable eddy currents);

(3) the resolution should be as high as may be attained (which requires that the slots in the code wheel be made very small).

By meeting the first two requirements, a fairly high signal-to-noise ratio, V_1/V_2, is obtained, so that 1's can reliably be differentiated from 0's.

According to requirement (2), the material of the code wheel should be such that appreciable eddy currents can exist in it. For

this to happen, the material should be readily penetrable by electromagnetic waves.

The penetration depth is a function of the resistance and relative permeability of the wheel material. The smaller the penetration depth, the thinner the wheel should be made, and the narrower the air gap can be made. From this point of view, the best materials for code wheels are Permalloy, electrical-sheet steel, and silicon iron. A code wheel fabricated from one of these materials will give the desired attenuation of the output signal at a thickness of about 0.1 mm. However, this thickness coupled with a great number of slots will not give the wheel the necessary stiffness. Calculations show that the code wheel will have the necessary stiffness at a thickness of 0.5 to 0.6 mm. With this thickness, however, code wheels may well be made from the less scarce materials such as copper and aluminium. This is why code wheels intended for induction reading are now mainly made from the last two materials.

With a code wheel of acceptable dimensions, the desired high resolution can be obtained by making sufficiently small slots in the wheel, having a width S_w at mean circumference and a radial length l_w. These dimensions should be properly matched to those of the cores, S_c and l_c, and the distance between the core poles, l_p. As a rule, it is sought to satisfy the following inequalities:

$$S_c < S_w$$
$$(l_p + l_c) < l_w$$

It is important to reduce the core size because the main flux shown in Fig. 8.45 and the opposing flux due to eddy currents in the wheel tend to buckle in the air gap. If the cores and slots were made equal or nearly equal in size, this buckling would markedly reduce the '1' signal in the secondary winding, W_2, which is highly undesirable. To make this signal sufficiently strong, the cores should be placed as closely together as practicable. The wheel teeth which represent binary 0's would likewise be larger in size than the core section in order to intercept the main flux and to attenuate the '0' signal. It is this buckling of the flux that controls the difference in size between the cores and the air gap. The minimum acceptable size of the cores and air gap determines, in the final analysis, the minimum size of slots in the code wheel and, as a consequence, its resolving power. For the same reason, open slots (in the edge of the wheel) may

be made somewhat smaller than closed slots, with all other conditions being equal.

Apart from code wheels, A/D converters may use code cylinders. In the latter, the slots may be made open.

The cores for the primary and secondary windings are fabricated from materials having high permeability and saturation induction. These properties are manifested by ferromagnetics with a narrow hysteresis loop — the narrower and the steeper the loop, the higher the permeability of the material. Permeability may be

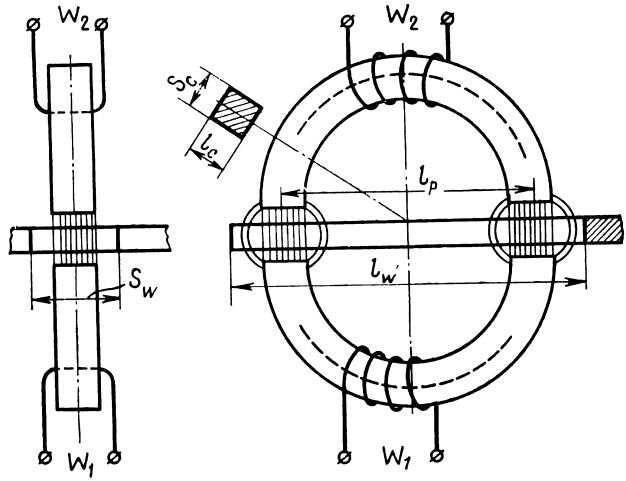

Fig. 8.45. Magnetic circuit

as high as 10^5 or 10^6 gausses/oersted for ferromagnetic metals and 10^3 or 10^4 gausses/oersted for nonmetallic ferromagnetics. The minimum cross-section of a core from such a material, having sufficient mechanical strength, is about 1 mm^2. Assuming that slots and lands are 1.5 mm wide, the code wheel of an eight-digit converter will be somewhat greater than 100 mm in diameter. This is the limit of a code wheel for converters using induction reading.

8.8. OPTICAL GRATINGS

Highly accurate analog-to-digital conversion can be achieved with *optical gratings*. A grating is a combination of light-transmitting and light-absorbing stripes, or slits.

When slits are spaced widely apart, no diffraction or interference phenomena occur. These coarse gratings are ruled with less than 20 slits to the millimetre.

Where slits are ruled closer together, a 'fine' diffraction grating results in which the passage of light is accompanied by diffraction and interference.

In A/D converters, gratings may be combined, or coupled in one of four ways, described as shutter coupling, grid coupling, vernier coupling, and moire coupling.

Shutter-coupled gratings. In this case, the shaft whose angular position is to be converted to a number mounts a code wheel on which slits are ruled in radial directions. Mounted coaxially with it is another stationary wheel (or part thereof). There is a light source on one side of the wheels, and a light detector on the other. As the movable wheel rotates, its slits line up alternately with the slits and opaque lands on the stationary wheel. In the former case, a maximum luminous flux will pass, while in the latter, it will be a minimum. A complete cycle of changes in the luminous flux from a maximum to a minimum occurs as the shaft rotates through one slit spacing and is repeated during the next.

Grid-coupled gratings. In this case, the light detector can intercept light from several areas within its field of vision at a time. This arrangement can collect a considerable amount of light even though the individual areas are very small, and balance out the inaccuracies in slit ruling.

Vernier-coupled gratings. There are two wheels arranged coaxially with each other, one movable and the other stationary. The movable wheel is mounted on the shaft whose angular position is to be converted to digital form. In contrast to shutter-coupled wheels, one of the vernier-coupled wheels has fewer slits (say, 10% fewer) than the other. When the two wheels are illuminated by a light source, the luminous flux varies cyclically from a maximum (when the slits on the movable wheel line up with those on the stationary wheel) to a minimum (when the slits on the movable wheel line up with the opaque lands on the stationary one). This cycle occupies a zone which embraces several slits; that is, the zone spacing is greater than the slit spacing. As the wheel moves through a distance equal to one slit spacing, the zone moves by an amount equal to the zone spacing, thereby producing a scale-up equal to the ratio of zone to slit spacing. The occurrence of a zone (its dark area) is sensed by a light detector.

The scale-up effect enables the angular position of the shaft to be monitored in fractions of a slit spacing, thereby enhancing the accuracy of conversion.

Moire gratings. Moire is an optical effect arising when one set of parallel lines is superimposed on another set so that the lines of the two sets intersect at acute angles. In doing so, they produce a pattern of alternating light and dark lines at right angles to the bisectors of these acute angles. If the lines in the superimposed sets are radial, the moire pattern will present concentric circles. The width of moire bands mainly depends on the angle at which the two sets of lines intersect. As the angle increases, the bands become progressively narrower until they disappear altogether at an angle of over 30°.

When one set of lines is moved relative to the other, the moire pattern moves too, but through a distance tens or even hundreds of times greater. This scale-up of displacement leads to a high accuracy of shaft-to-number conversion. When the sets of lines move (relative to each other) through a distance equal to line spacing $w_1 = w_2 = w$ (which is the same in both sets), the moire band moves through a distance equal to its spacing, W

$$W = kw$$

where k is the scale-up factor due to the moire effect.

For the case under discussion,

$$k = 1/\left(2 \sin \frac{\alpha}{2}\right)$$

where α is the angle between the sets of lines.

With a sufficiently small α,

$$k = 1/\alpha$$

A photoelectric system can sense the position of the moire band by an amplitude or a phase method.

Amplitude method. With this method, the position of the moire band is determined from the amount of light reaching light detectors installed along its travel within a single band spacing. If it is desired to observe the moire band over a distance Δ, then the number n of light detectors needed within the band spacing will be

$$n = W/\Delta$$

This method is a modification of incremental conversion, with all its merits and demerits.

Phase method. With this method, four light detectors, *LD1*, *LD2*, *LD3* and *LD4*, are equidistantly arranged within the moire band spacing, *W*. Figure 8.46*a* shows the relative positions of the raster gratings when the phase shift φ of the moire band relative to some fixed datum is 0, 30, 45, 60 and 90°. Figure 8.46*b* shows variations in the illumination within the band spacing for the various positions of the moire pattern, and Fig. 8.46*c* shows the photocurrents of the light detectors corresponding to each level of illumination. The illumination is a minimum at the middle of the moire band. As the moire band is shifted, the illumination of the light detectors changes, and the signals they produce also change. The signals are picked off the light detectors during four clock times *A*, *B*, *C* and *D* (Fig. 8.46*d*) so that the signal produced by *LD1* is sensed at clock times *A* and *B*, by *LD2* at clock times *B* and *C*, by *LD3* at clock times *C* and *D*, and by *LD4* at clock times *D* and *A*. As is seen, every next signal overlaps the previous one for a quarter of a cycle ($\pi/2$ radians). This sequence is maintained by the *AND1*, *AND2*, *AND3* and *AND4* gates, and control flip-flops, *FF1*, *FF2* and *FF3* (Fig. 8.47). When a clock pulse is applied to the T-input of the flip-flop *FF1* a control pulse is generated for the succeeding flip-flops, *FF2* and *FF3*. Their outputs deliver square pulses which open the *AND* gates for signals generated by the light detectors. These signals go to a summing amplifier, *SAmp*, past which the a.c. signal component is integrated by an integrating amplifier, *IAmp*. The signal waveforms at the inputs to *SAmp* and *IAmp* for different positions of the moire pattern are shown in Fig. 8.46*e* and *f*. The phase shift φ of the signal appearing at the output of the integrator relative to a reference signal (which may be the control signal of the first *AND* gate) is measured by a comparison circuit, *Comp*. The comparison circuit generates a control signal which is applied to the input of *FF4* when the integrated curve passes through zero.

The flip-flop *FF4* controls the *AND5* gate which passes timing pulses on to the counter, *Count*. When a cycle of measurement is commenced, that is, when a control signal is applied to *AND1*, *FF1* changes state and opens *AND5* for timing pulses. At the end of the cycle, that is, when the integrated curve passes through zero, no more timing pulses are allowed to reach the counter. Obviously, the pulses accumulated by the counter will be proportional to the time phase shift φ.

It can be shown that the time phase shift φ is a linear function of the moire-pattern displacement, Ω.

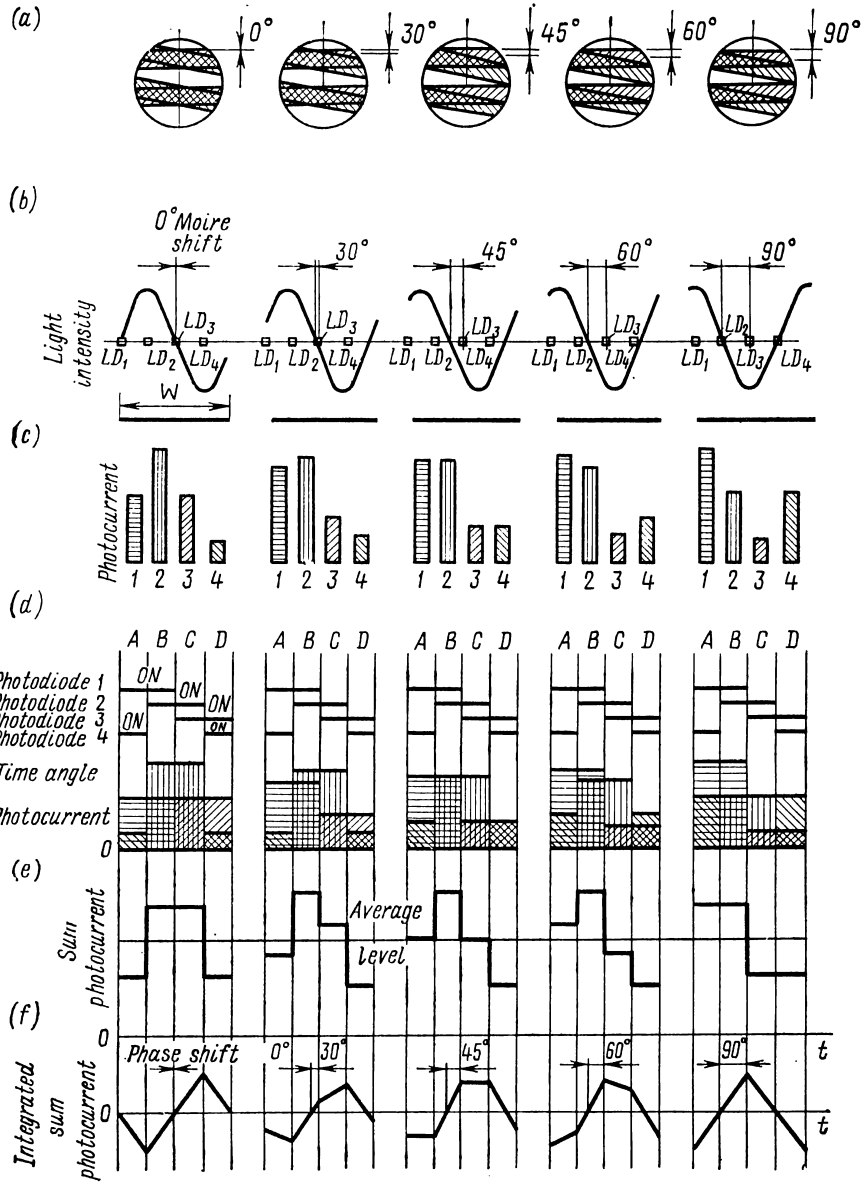

Fig. 8.46. Moire-grating converter: moire patterns and timing diagrams

If the distribution function of the luminous flux within the moire band spacing is a symmetric triangle, then the currents of the light detectors are

$$\left.\begin{aligned} i_1 &= I\,[1 + mg\,(\Omega)] \\ i_2 &= I\,[1 + mg\,(\Omega + \pi/2)] \\ i_3 &= I\,[1 + mg\,(\Omega + \pi)] \\ i_4 &= I\,[1 + mg\,(\Omega + 3\pi/2)] \end{aligned}\right\} \qquad (8.21)$$

where $g(\Omega)$ is a function describing the triangular distribution of the luminous flux within the moire band spacing, and m is the depth of modulation.

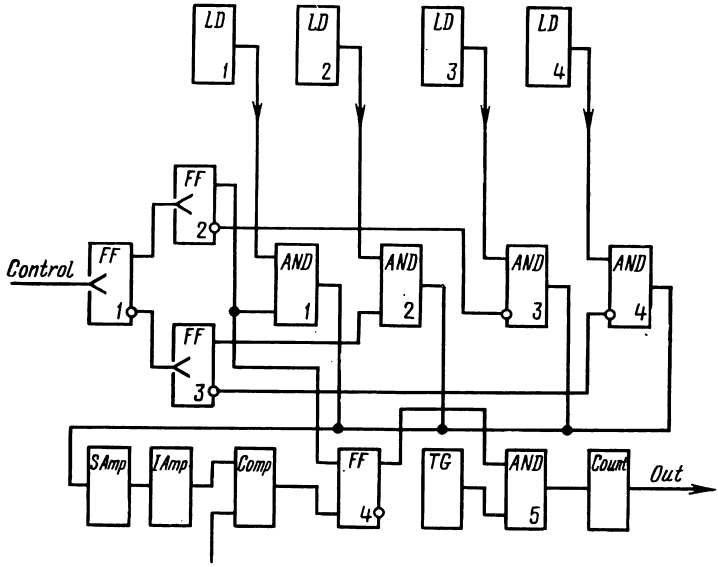

Fig. 8.47. Block diagram of phase reading

The amount of luminous flux, Φ, reaching a light detector depends on the area of the light parallelogram (moire pattern), S, intercepted by the sensing slot.

Obviously, S is a function of the position x of the moire pattern relative to the sensing slot. If the lines in the two rasters are spaced the same distance apart, $w_1 = w_2$, and the slits and lands are of the same width, $S(x)$ is a triangular function.

The dependence of the luminous flux $\Phi(x)$ on $S(x)$ may be written as

$$\Phi(x) = ES(x)$$

where E is the illumination of the moire-pattern field.

When illumination is constant, the plot of light flux as a function of the position of the moire pattern likewise yields a triangle. Accordingly, if the current of a light detector is a linear function of the luminous flux, its distribution will likewise be triangular.

Since the currents i_1, i_2, i_3 and i_4 of the light detectors are sampled at different clock times, i_1 and i_3 never appear together at the input to the amplifier, which is also true of i_2 and i_4. However, the sum of i_1 and i_3 is present at the same time as the sum of i_2 and i_4 is. These pairwise sums are shifted from each other by a quarter of a cycle, that is, $\pi/2$. The alternating component of the resultant signal may be looked upon as the sum of terms proportional to the difference signals, $i_1 - i_3$ and $i_2 - i_4$, shifted through $\pi/2$ from each other.

Recalling Eq. (8.21), the terms of the alternating component may be written as

$$i_1 - i_3 = I\,[1 + mg(\Omega)] - I\,[1 + mg(\Omega + \pi)]$$

Since, however, $g(\Omega + \pi) = -g(\Omega)$, then

$$i_1 - i_3 = 2mIg(\Omega) \tag{8.22}$$

Similarly,

$$i_2 - i_4 = I\,[1 + mg(\Omega + \pi/2)] - I\,[1 + mg(\Omega + 3\pi/2)]$$

Since, however, $g(\Omega + 3\pi/2) = -g(\Omega + \pi/2)$, then

$$i_2 - i_4 = 2mIg(\Omega + \pi/2) \tag{8.22a}$$

Each of the two differences changes in a symmetric rectangular fashion within each moire-band spacing.

Since the sum of the difference signals is then applied to an integrator, and the integral of a sum is equal to the sum of the integrals of the summands, the process of integration may be examined independently for each difference.

The integral of a function obeying a symmetric square distribution leads to a function obeying a symmetric triangular distribution. As already shown, the square functions are shifted from each other by $\pi/2$, therefore the triangular functions obtained by integration are likewise shifted by $\pi/2$ from each other. As a con-

sequence, the integrated sum signal for a sequence of moire bands recurring at a frequency ω has the form

$$i(\omega t) = (i_1 - i_3)\, g(\omega t) + (i_2 - i_4)\, g(\omega t - \pi/2)$$

where $g(\omega t)$ is a symmetric triangular function of time.

Recalling Eqs. (8.22) and (8.22a), we get

$$i(\omega t) = 2mIg(\Omega)\, g(\omega t) + 2mIg(\Omega + \pi/2)\, g(\omega t - \pi/2) \quad (8.23)$$

From analysis of the above expression it is seen that the function $i(\omega t)$ passes through zero at

$$\omega t = (2n + 1)\pi/2 - \Omega$$

Thus, the phase φ of the signal $i(\omega t)$ is a linear function of Ω. Since, however, the displacement of a moire band is k times the displacement, θ, of the grating, that is, $\Omega = k\theta$, the phase of the signal is a linear function of θ as well.

A grating system utilizing the moire effect can convert mechanical displacements not exceeding the slit spacing, $0 \leqslant \theta \leqslant w$.

If the displacement exceeds w and, in the general case, covers 360°, the moire system should be combined with a coarse A/D converter implementing any one of the above methods. In such a combination, the moire system will act as a fine converter.

8.9. VOLTAGE-TO-TIME-TO-DIGITAL CONVERTERS

This is an indirect voltage-to-digital conversion technique because the voltage of interest is first converted to a time interval, and the time interval is then converted to a number.

The technique may be implemented in several ways. The most commonly used ones are the ramp compare method and the staircase compare method.

Ramp compare. A ramp compare converter operates as follows (Fig. 8.48). Starting at time t_i which marks the commencement of a conversion cycle, the voltage V_{in} to be converted is compared with a reference ramp voltage, V_r. Upon detection of coincidence between the input signal and the ramp function, a pulse is generated which stops the cycle and also marks the end of the time interval, Δt_i, during which the ramp function rose to the magnitude of the input analog voltage. Since the ramp voltage is a linear time function within the conversion cycle, that is:

$$V_r = k\,\Delta t \quad (0 \leqslant \Delta t \leqslant \Delta t_{max})$$

and the value of k is known, the time interval Δt_i during each conversion cycle gives a univalued measure of the input analog signal, V_{in}:

$$V_{in,\,i} = V_{r,\,i} = k\,\Delta t_i$$

The time interval Δt_i is measured by counting clock pulses that occur within that interval. This is done by a binary counter, *Count*.

A simplified diagram of a voltage-to-time-to-digital converter is shown in Fig. 8.49. It consists of a highly-stable pulse generator, *PG*, an *AND* gate, a counter, *Count*, a control flip-flop, *FF*,

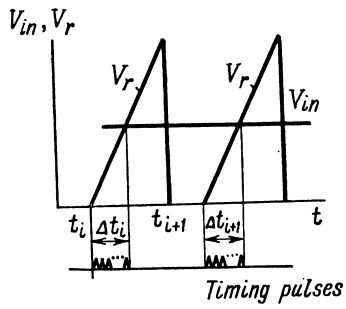

Fig. 8.48. Ramp-compare plot

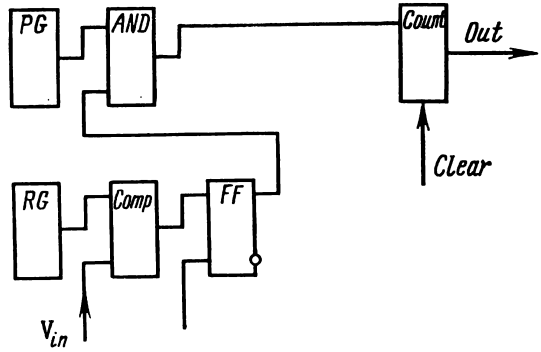

Fig. 8.49. Block diagram of a ramp-compare converter

a comparator, *Comp*, and a ramp generator, *RG*. Prior to a conversion cycle, a pulse is applied over the "*Clear*" line to clear the counter. As the ramp starts, the flip-flop *FF* is caused to change state so that it enables the *AND* gate for pulses to reach the counter. As soon as the input analog signal, V_{in}, is the same as the ramp, the comparator generates a signal which resets the flip-flop and disables the *AND* gate, so that no more pulses can reach the counter. The number n of pulses left in the counter will be proportional to the time interval Δt_i:

$$n = f_t\,\Delta t_i$$

where f_t is the repetition frequency of timing pulses.

The binary number left in the counter can then be sampled at the output in a parallel or serial code. This completes the conversion cycle. The counter is cleared and a pulse is generated to initiate the next cycle.

The duration of a conversion (or sampling) cycle, Δt_c, is chosen according to the rate of change of the input analog signal, V_{in}, such that during time Δt_c the change in V_{in} will not exceed the least significant digit in the binary number left in the counter. The maximum amplitude of the ramp, $V_{r\,\max}$, should be at least equal to the maximum permissible value of V_{in}. Hence, the rate of change of the ramp, $V_{r\,\max}/\Delta t_c$, is decided by the precision sought.

The limit of accuracy of such converters is dependent on the accuracy of the comparator which senses the occurrence of coincidence between V_{in} and V_r, and by the stability and linearity of the ramp.

RAMP GENERATOR. A simple ramp generator may be an amplifier with an *RC*-network at the output. When a trigger pulse is applied to the amplifier input, the voltage at the output of the *RC*-network rises exponentially at a time constant $\tau = RC$:

$$e_{out} = E_C(1 - e^{-t/\tau}) \quad (0 \leqslant t \leqslant \Delta t_c)$$

With some simplification, the initial portion of the exponential curve may be assumed to be linear. The error arising from this assumption may be determined by expanding the expression for the output voltage into a power series:

$$e_{out} = E_C\left[1 - 1 + \frac{t}{\tau} - \frac{1}{2!}\left(\frac{t}{\tau}\right)^2 + \frac{1}{3!}\left(\frac{t}{\tau}\right)^3 - \ldots\right]$$
$$= E_C\frac{1}{\tau}\left[1 - \frac{1}{2}\left(\frac{t}{\tau}\right) + \frac{1}{6}\left(\frac{t}{\tau}\right)^2 - \ldots\right]$$

The output voltage rising at the same slope as the original ramp is

$$e'_{out} = E_C t/\tau$$

Thus, the error (Fig. 8.50) due to the nonlinearity of the exponential curve is

$$\varepsilon = e'_{out} - e_{out} + E_C\frac{t}{\tau} - E_C\frac{t}{\tau}\left[1 - \frac{1}{2}\left(\frac{t}{\tau}\right) + \frac{1}{6}\left(\frac{t}{\tau}\right)^2 - \ldots\right]$$
$$= E_C\frac{t}{\tau}\left[\frac{1}{2}\cdot\frac{t}{\tau} - \frac{1}{6}\left(\frac{t}{\tau}\right)^2 + \ldots\right]$$

At $t/\tau \ll 1$, within the initial portion of the exponential curve

$$e_{out} \approx E_C t/\tau$$

and so the error is zero very nearly. As Δt_{max} is increased, the error increases too.

The circuit just examined suffers from a relatively low accuracy and can handle a low input analog voltage. Accuracy may be enhanced through the use of a feedback ramp generator. In such a generator, the current through the capacitor is maintained constant. As will be recalled, the voltage across, and the current

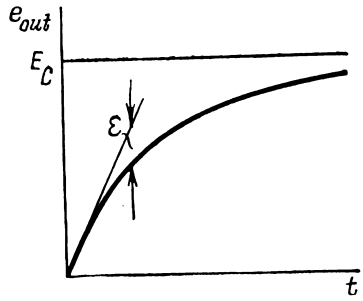

Fig. 8.50. Error due to the nonlinearity of an exponential function

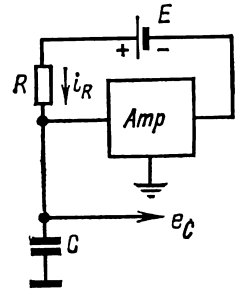

Fig. 8.51. Circuit of a feedback ramp generator

through, a capacitor, e_C and i_C, respectively, are connected by the following relation:

$$e_C = \frac{1}{C}\int_0^t i_C\,dt$$

As is seen, at i_C = constant, the voltage is a linear function of time:

$$e_C = (i_C/C)\,t$$

where i_C/C = constant.

The circuit in which the current through the capacitor is held constant is shown in Fig. 8.51. This is done by a feedback amplifier, *Amp*, with a gain k.

The currents through the capacitor C and the resistor R are

$$i_C = C\,de_C/dt$$
$$i_R = (E + ke_C)/R - e_C/R$$

Assuming that the input resistance of the feedback amplifier and the load resistance are infinite, while the output voltage of

the amplifier is zero, we get

$$i_C = i_R$$

Substituting the expressions for the currents in the above equation gives

$$C\,de_C/dt = (E + ke_C)/R - e_C/R$$

and, after some manipulation

$$de_C/dt + e_C\,(1 - k)/RC = E/RC$$

On setting the discharge time constant $\tau = RC$, we finally get

$$de_C/dt + e_C\,(1 - k)/\tau = E/\tau$$

Assuming zero initial conditions, the solution of the above differential equation may be written as

$$e_C = E/(1 - k)\,\{1 - \exp - \{[(1 - k)/\tau]\,t\}\}$$

This expression may be expanded into a power series as follows:

$$e_C = \frac{E}{1-k}\left[(1-k)\frac{t}{\tau} - \frac{(1-k)^2}{2!}\frac{t^2}{\tau^2} + \frac{(1-k)^3}{3!}\frac{t^3}{\tau^3} - \cdots\right]$$

$$= \left[E\frac{t}{\tau} - \frac{1-k}{2!}\frac{t^2}{\tau^2} + \frac{(1-k)^2}{3!}\frac{t^3}{\tau^3} - \cdots\right]$$

This equation shows that if the amplifier gain is $k = 1$, the capacitor voltage rises linearly

$$e_C = Et/\tau$$

With the amplifier gain held at $k = 1$, the accuracy with which e_C can be maintained a linear function of time will depend on the stability of the supply source, E, and the circuit parameters R and C.

An amplifier of unity gain may be a cathode or emitter follower. Each time the ramp is to be started, the capacitor should be shunted by an electronic switch so that the ramp can start from zero.

Staircase compare. A staircase-compare A/D converter operates in much the same manner as a ramp compare circuit. The basic difference is that the ramp type function now consists of a series

of small step functions (Fig. 8.52), each writing a binary 1 in the counter. By the time the input analog voltage, V_{in}, and the staircase reference voltage, V_r, become equal, the number left in the counter is proportional to the amplitude of V_{in}. The circuit of a staircase-compare converter is shown in Fig. 8.53.

Initially, the counter, *Count*, is set to zero and the *AND* gate is disabled. When a conversion cycle begins, the pulse generator, *PG*, is started, the flip-flop, *FF*, changes state and gates out timing pulses to the counter. At the same time, a staircase generator, *SG*, adds a step to the ramp for each timing pulse. This

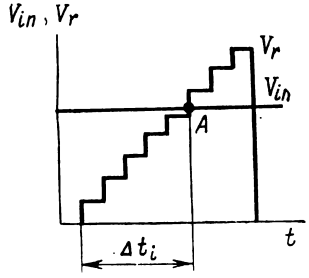

Fig. 8.52. Staircase-compare plot

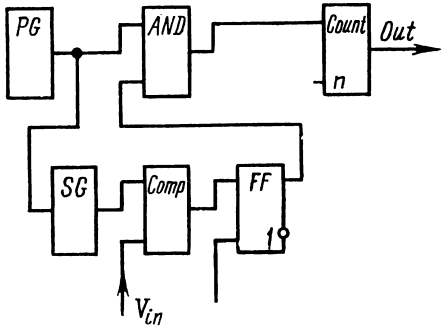

Fig. 8.53. Block diagram of a staircase-compare converter

sequence continues until the staircase reference voltage V_r is the same as the input analog voltage, V_{in}. The instant when the two voltages coincide is sensed by a comparator, *Comp*. Upon detection of coincidence, the comparator generates a pulse which resets the flip-flop and disables the *AND* gate, so that no more timing pulses can reach the counter. After the contents of the counter are sampled, the circuit is reset in preparation for taking the next measurement.

The circuit of a staircase generator is shown in Fig. 8.54. The generator input accepts positive rectangular pulses with amplitude *E*. These pulses charge the capacitors *C*1 and *C*2. When the input pulse ceases, the capacitor *C*1 retains the charge, while the capacitor *C*2 begins to discharge through diode *D*2.

The anode of the diode is coupled to output via a cathode follower, *CF*, with a gain close to unity. As a result, the capacitor *C*2 recharges in opposite polarity to a voltage *e* (where *e* is output voltage).

Neglecting the forward resistance of the diode, the incremental step added to the staircase waveform each time the counter is pulsed may be expressed as

$$\Delta e = \frac{C_2}{C_1 + C_2}(E - e_{C2} - e)$$

where e_{C2} is the voltage across the capacitor $C2$

But, $e_{C2} = -e$, and so

$$\Delta e = \frac{C_2}{C_1 + C_2} E$$

It is seen that the incremental step in the staircase waveform has a constant magnitude, Δe, independent of the output voltage.

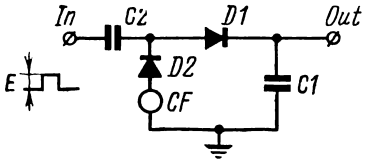

Fig. 8.54. Circuit of a staircase voltage generator

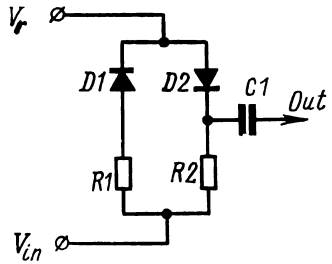

Fig. 8.55. Circuit of a diode comparator

Obviously, any departure of the cathode-follower gain from unity will upset the equality $e_{C2} = -e$, and every next step will differ in height from the previous one. This sets a limit to the accuracy of conversion.

Comparator circuits. The comparator which detects coincidence between the input analog voltage, V_{in}, and the reference voltage, V_r, may have a variety of configurations. The simplest one is that using diodes (Fig. 8.55). Until V_{in} and V_r become the same, the diode *D1* remains conducting and the diode *D2* non-conducting. When the two voltage become equal, *D1* turns off, *D2* turns on, and a current begins to flow through *R2*. As a result, a voltage pulse appears on the output line.

A major limitation of this configuration is low sensitivity. Also, it varies with the magnitude of the signals being compared because the volt-ampere characteristic of the diodes has a slowly rising portion from 0 to 200-300 mV. Furthermore, the reverse

resistance of the diode changes with temperature, and this cannot but affect the sensitivity of the circuit. For example, the reverse resistance of the Soviet-made type Д9К germanium diode drops to about one-thousandth as the temperature rises from +20 to +70 °C.

The magnitude of the output pulse that can be detected by the circuit depends on the sensitivity of the succeeding element (in our case, the control flip-flop) and on the difference between the voltages being compared. To bring down the magnitude of the

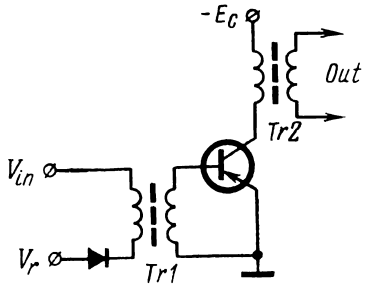

Fig. 8.56. Circuit of a transistor comparator

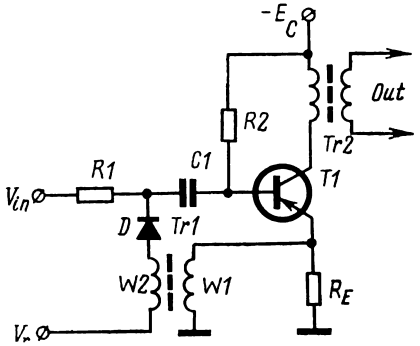

Fig. 8.57. Circuit of a positive-feedback comparator

output pulse, that is, to enhance the sensitivity of a diode comparator, it is customary to combine it with an amplifier or a relaxation stage. In such a combination, it is essential to synchronize the diode circuit and the following stage. One such comparator circuit combined with an amplifier based on a transistor is shown in Fig. 8.56. As long as V_{in} exceeds V_r, there is no current flowing in the primary winding, *W1*, of the pulse transformer, *Tr1*. As soon as V_r exceeds V_{in}, a signal appears in *W1*. The signal induced into the secondary winding, *W2*, is amplified and induced into the secondary winding of the output transformer, *Tr2*. This circuit can have a sensitivity of 50 to 100 mV.

Any further improvement in the sensitivity of the comparator may be attained by means of positive feedback between the comparator and the amplifier stage (Fig. 8.57), with the aid of a pulse transformer, *Tr1*. Initially, that is before the reference voltage exceeds the input analog voltage, there is practically no current flowing in the primary circuit, *R1* — *D*. At the same time, the transistor *T1* is held very close to the saturated state (or conduct-

ing) through an appropriate choice of *R2*. A direct current is flowing in the emitter circuit, and there is no positive feedback. As soon as V_r exceeds V_{in}, the diode *D* turns on, and a current begins to flow in the comparing circuit. Now a positive signal is applied to the base of transistor *T1*, causing it to move towards cut-off and its collector and emitter currents to decrease. The decrease in the emitter current induces in the secondary winding *W2* of the transformer *Tr1* a signal which drives the diode *D* towards conduction at a faster rate, thereby building up the positive potential at the base of *T1* and driving it to cut-off. This process proceeds cumulatively and at a very high rate.

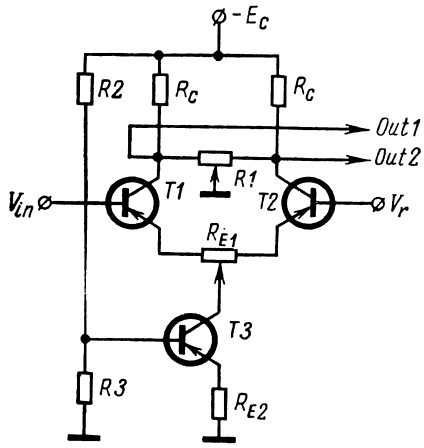

Fig. 8.58. Circuit of a preamplifier comparator

As the transistor *T1* changes from saturation to cut-off, the collector current suffers a change which causes a signal to be induced into the secondary winding *W2* of the pulse transformer *Tr2*.

Feedback is most effective when the gain of the feedback circuit is close to unity, which can be achieved by arranging the pulse transformer *Tr1* to have a turns ratio of $W2/W1 = 1.5$ or 2 to 1.

Positive feedback improves the sensitivity of the circuit by an order of magnitude in a sufficiently wide range of changes in the quantities being compared. With circuit parameters properly chosen, the sensitivity of the circuit may be made as high as 20 mV in the range 0-10 V.

The relaxation circuits that may be combined with the comparator are the Schmitt trigger, the multivibrator and the blocking oscillator.

High sensitivity is achieved in comparators utilizing signal preamplification. This arrangement is especially useful where low-level signals are compared. The basis of such circuits is a differential amplifier (Fig. 8.58). The input analog signal to be converted, V_{in}, is applied to the base of a transistor, *T1*, and the reference voltage, V_r, to the base of another transistor, *T2*. If the circuit is symmetrical, the signal appearing at the output ter-

minals of the differential amplifier will be proportional to the difference between the applied voltages:

$$V_{out} = k(V_{in} - V_r)$$

where k is the gain factor of the differential amplifier.

This circuit has found wide use because it can handle the difference voltage of any polarity, because the output voltage will then take an appropriate sign. The sensitivity of this circuit is only limited by the drift voltage and is higher than it is in other types of d.c. amplifier. This is achieved by balancing zero currents and voltages across the emitter-base junctions of transistors *T1* and *T2* and by inserting a third transistor, *T3*, to stabilize the total current of the differential amplifier. The low total current markedly reduces the drift due to changes in the gain β. The comparator using a differential amplifier may have a sensitivity of a few millivolts (in the temperature range 20-35 °C). The differential amplifier examined above can drive a high-ohmic load.

8.10. SELECTIVE-SUBTRACTION A/D CONVERTERS

For their operation, these converters depend on the fact that any voltage to be converted may be expressed as the sum of terms proportional to successive powers of 2:

$$V_{in} = x_n 2^n V_r + x_{n-1} 2^{n-1} V_r + \ldots + x_i 2^i V_r + \ldots + x_0 2^0 V_r$$

where V_{in} is the voltage to be converted, x_i may take the value 0 or 1, and $2^i V_r = V_{r,i}$ is the weighted reference voltage.

A selective-subtraction A/D converter operates by ascertaining whether or not the input analog voltage contains some weighted reference voltage, $2^i V_r$. If it does, a 1 is written in the respective position of the binary number x_i at the converter output; if it does not, a 0 is written.

The highest-order weighted voltage should be greater than half the maximum value of the input analog voltage, that is:

$$2^n V_r > V_{in\,max}/2$$

or

$$2^{n+1} V_r > V_{in\,max}$$

The magnitude of the analog voltage is balanced against weighted reference voltages by comparison and subtraction, proceeding from the most significant value to the least. As the first

step, V_{in} is compared with the highest-order reference voltage, V_r. If V_{in} exceeds or is equal to V_r, a 1 is written in the most significant digit position of the output binary number, and the difference $V_{in} - 2^n V_r$ is determined. If V_{in} is less than V_r, a 0 is written in the most significant digit position of the output binary number, and the difference is not determined.

During the next step, the next weighted reference voltage is balanced against the last-found difference, $V_{in} - 2^n V_r$ if $x_n = 1$, or the total input voltage if $x_n = 0$. Again, if $(V_{in} - 2^n V_r) \geqslant \geqslant 2^{n-1} V_r$ or $V_{in} \geqslant 2^{n-1} V_r$ (at $x_n = 0$), a 1 is written in the next

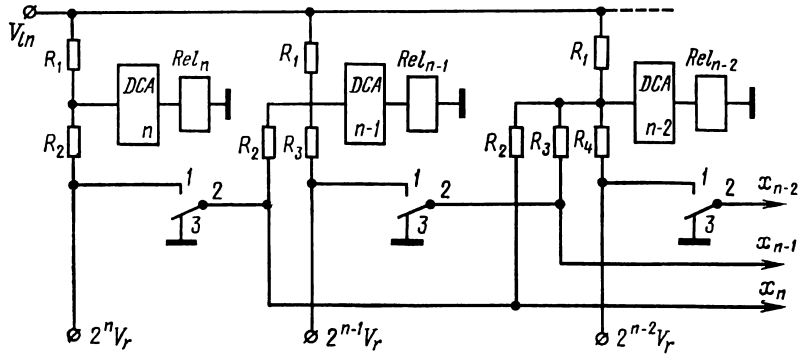

Fig. 8.59. Circuit of a selective-subtraction A/D converter

less significant digit position of the output binary number and the difference $(V_{in} - 2^n V_r) - 2^{n-1} V_r$ is determined. If $(V_{in} - 2^n V_r)$ is less than $2^{n-1} V_r$, a 0 is written in the next less significant digit position, x_{n-1}, of the output binary number, and the difference is not determined. Then comes the third step, and the procedure is repeated until all digit positions are found.

The three stages handling the most significant digits on the principle just explained are shown in the circuit of Fig. 8.59. It consists of d.c. amplifiers, DCA_i, two-position polarized relays, *Rel*, biased to make the lower contact, and weighted reference voltage sources, $2^i V_r$.

The input of the amplifier, DCA_n, handling the most significant value, accepts the difference between the analog voltage, V_{in}, and the most significant reference voltage, $2^n V_r$. If the difference is positive, a signal appears at the amplifier output, causing the polarized relay to pick up (it moves towards contact *1*). As a result, a 1 signal is generated at the x_n output, and the weighted

voltage $2^n V_r$ is passed on to the next compare-and-subtract stage. The input of the amplifier, DCA_{n-1}, handling the next less significant value, accepts the difference between V_{in} and the sum of two most significant reference voltages $(2^n V_r + 2^{n-1} V_r)$. If the difference is positive, a signal appears at the amplifier output, causing the polarized relay Rel_{n-1} to operate, a 1 to be generated at the x_{n-1} output, and the weighted voltages to be passed on to the next compare-and-subtract stage.

If the difference, $V_{in} - 2^n V_r$, applied to the most significant compare stage is negative, the polarized relay will not operate, switch *2* remains at contact *3*, a 0 will be generated at the x_n output, and the weighted voltage $2^n V_r$ will not be passed on to the next compare stage. In the next stage, the input voltage will then be only compared with the next lower weighted voltage, etc.

The precision of such a converter and, as a consequence, the digit range of conversion are decided by the stability of weighted reference voltages, sensitivity of the d.c. amplifiers, and operation of the polarized relays. For higher speed and better reliability, the polarized relays may be replaced by electronic flip-flops.

Instead of a multiplicity of weighted reference voltages, only one may be used, if a greater number of d.c. amplifiers be incorporated. The rationale of such an arrangement lies in the fact that the comparison and subtraction steps examined above:

$$\begin{aligned}
&\text{step 1:} && V_{in} - 2^n V_r = \Delta V_0 \\
&\text{step 2:} && \Delta V_0 - 2^{n-1} V_r = \Delta V_1 \\
&\text{step 3:} && \Delta V_1 - 2^{n-2} V_r = \Delta V_2 \\
&\quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
&\text{step } (n+1): && \Delta V_{n-1} - 2^0 V_r = \Delta V_n
\end{aligned}$$

may be replaced as follows:

$$\begin{aligned}
&\text{step 1:} && V_{in} - 2^n V_r = \Delta V_0 \\
&\text{step 2:} && 2^1 (\Delta V_0 - 2^{n-1} V_r) = 2^1 \Delta V_1
\end{aligned}$$

or

$$\begin{aligned}
&2^1 \Delta V_0 - 2^n V_r = 2^1 \Delta V_1 = \Delta V_1' \\
&\text{step 3:} \quad 2^2 (\Delta V_1 - 2^{n-2} V_r) = 2^2 \Delta V_2
\end{aligned}$$

or

$$2\Delta V_1' - 2^n V_r = 2^1 \Delta V_2 = \Delta V_2'$$

. . . . . .

$$\text{step } (n+1)\text{: } 2^n(\Delta V_n - 2^0 V_r) = 2^n \Delta V_n$$

or

$$2\Delta V_{n-1} - 2^n V_r = 2^n \Delta V_n = \Delta V_n'$$

As is seen, each step may use the same weighted voltage equal to the most significant value, $2^n V_r = V_r'$. Then at all steps, except the first, instead of the voltage to be converted, comparison should be made with twice the difference from the previous step, $2\Delta V_{i-1}'$. The above equations hold for the case where a 1 is written in each digit position of the output binary number. In the general case, assuming that a 0 may appear in the digit positions x_i

$$x_i = \begin{cases} 0 & \text{if} \Delta V_i' < V_r' \\ 1 & \text{if} \Delta V_i' \geqslant V_r' \end{cases}$$

For the ith step:

$$\Delta V_i' = 2(\Delta V_{i+1}' - V_r' x_{i+1})$$

where $V_r' = 2^n V_r$.

As an example, examine a five-stage converter designed to cover a range of input voltages from 0 to 30 V. The quantization step is $V_r = 1$ V. The reference voltage is $2^{5-1} V_r = 16$ V.

The input analog voltage will be converted into a binary number $x_4 x_3 x_2 x_1 x_0$. If $V_{in} = 21$ V and $x_4 = 1$, then, by comparing V_{in}, $\Delta V_3'$, $\Delta V_2'$, $\Delta V_1'$ and $\Delta V_0'$ with the reference voltage V_r', we will find the respective digits of the output binary number to be:

$$\Delta V_3' = 2(21 - 16 \times 1) = 10 < 16 \qquad x_3 = 0$$

$$\Delta V_2' = 2(10 - 16 \times 0) = 20 > 16 \qquad x_2 = 1$$

$$\Delta V_1' = 2(20 - 16 \times 1) = 8 < 16 \qquad x_1 = 0$$

$$\Delta V_0' = 2(8 - 16 \times 0) = 16 \qquad x_0 = 1$$

that is, the output binary number is 10101.

The three most significant stages of a converter implementing the above rules are shown in Fig. 8.60. Each stage includes a subtraction circuit, *SC*, and a circuit, *DC*, which doubles the difference. If a comparison step shows that the difference is positive,

$\Delta V_i' \geqslant V_r'$, a 1 is written in the ith digit position of the output binary number x_i, and the difference is doubled before it is applied to the next stage for comparison with V_r'. If a comparison step shows that the difference is negative, $\Delta V_i' < V_r'$, a 0 is written in the ith digit position of the output binary number x_i, and the difference is passed on to the next stage without being doubled.

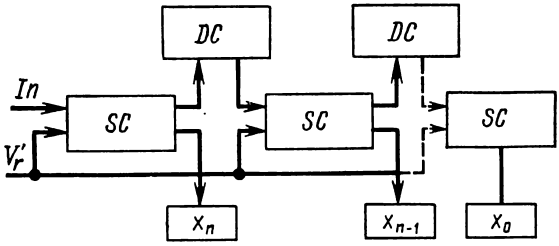

Fig. 8.60. Block diagram of a doubled-difference converter

8.11. FEEDBACK A/D CONVERTERS

The key element of a feedback A/D converter is a digital-to-analog converter which generates voltages proportional to the number being matched. If the voltage thus generated is equal to the input analog voltage, the binary number thus matched is the sought result.

The circuit of a feedback A/D converter is shown in Fig. 8.61. Consider its operation in some detail (Fig. 8.62).

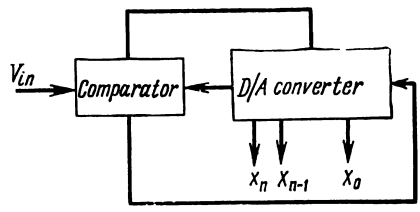

Fig. 8.61. Block diagram of a feedback converter

The input analog voltage to be converted is compared with a voltage generated by a digital-to-analog converter. This voltage is a sum of reference voltages whose weights correspond to those of the various digits in the output binary number.

The sum of reference (or weighted) voltages is built up, proceeding from the most significant value, until it matches the input analog voltage accurate to the least weighted voltage. When the input voltage is found to contain a given weighted voltage, $V_{r,\,i}$, a 1 is written in the ith digit position of the output binary number; if not, a 0 is written.

The channel commutator steers V_{in} to a comparator, *Comp*, the other input of which accepts reference voltages generated by a restistance network decoder, *RND*, when appropriate branches are energized with voltages coming from flip-flops, *FF1* through *FFn*. Clock pulses cause the flip-flops to change state and apply these voltages serially, starting with the branch assigned the highest weight. The clock pulses are generated and applied to the

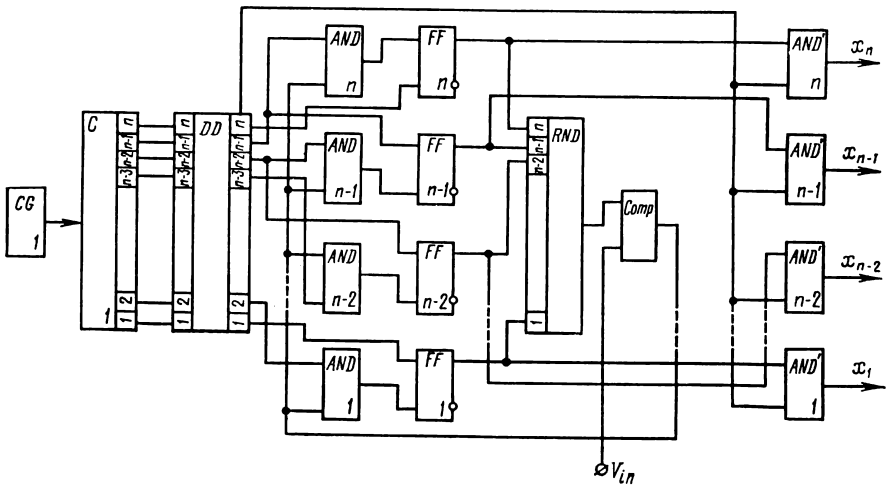

Fig. 8.62. Feedback converter

respective flip-flops by a clock generator, *CG*, via a binary counter, *C*, and a diode decoder, *DD*.

When the first clock pulse arrives, the first flip-flop changes state, and the weighted resistance network, *RND*, generates a voltage assigned the highest weight, $V_{r,\,n}$. This voltage is applied to the comparator where it is compared with V_{in}. If $(V_{in} - V_{r,\,n}) < 0$, the comparator generates a signal which controls the chain of *AND* gates, *AND1* through *ANDn*. If $(V_{in} - V_{r,\,n}) > 0$, no control signal is generated.

The second clock pulse causes the second flip-flop to change state, the next weighted branch of the resistance network to be energized, and the next weighted voltage, $V_{r(n-1)}$, to appear at its output. At the same time, the second clock pulse is applied to *AND1*. If the other input of *AND1* accepts a signal from the comparator, that is, if $(V_{in} - V_{r,\,n}) < 0$ the second clock pulse is allowed to pass on to the second input of *FF1* and resets it, the-

reby removing the voltage from the respective weighted branch of the resistance network. If $(V_{in} - V_{r,\,n}) > 0$ the flip-flop remains "set". Thus, during the second clock time the comparator compares the input voltage either with the second lower weighted voltage, $V_{in} - V_{r,\,n-1}$, or with the sum of two weighted voltages, $V_{in} - (V_{r,\,n} + V_{r,\,n-1})$, depending on the outcome of the previous comparison. At the same time, the sign of the difference is determined, etc.

The waveforms of the weighted voltages representing a five-digit output binary number are shown in Fig. 8.63. The voltage of weight *1* is $V_{r,\,n}$, that of weight *2* is $V_{r,\,n-1}$, that of weight *3* is $V_{r,\,n-1} + V_{r,\,n-2}$, that of weight *4* is $V_{r,\,n-1} + V_{r,\,n-3}$, and that of weight *5* is $V_{r,\,n-1} + V_{r,\,n-3} + V_{r,\,n-4}$. At the end of a complete cycle occupying n clock periods, the flip-flops will be either set or reset, according as the final sum of reference voltages contains the respective weighted voltages or not. The flip-flop outputs are connected in parallel with the weighted branches of the resistance network to the control inputs of the output gates, *AND'1* through *AND'n* (see Fig. 8.62). After n clock times of comparison, the counter sends out a read pulse via the diode decoder, *DD*, to the inputs of all *AND'* gates. The *AND'* gates whose flip-flops store 1's will gate out this pulse on the output lines. Thus, after $n + 1$ clock times of a conversion cycle, the input analog voltage appears at the output in a parallel binary code.

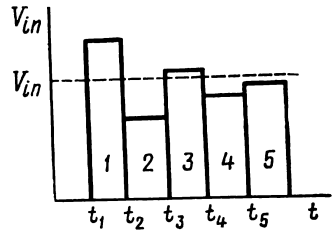

Fig. 8.63. Waveforms of output voltages across a resistance network

Feedback A/D converters are fast and accurate and have found wide use.

8.12. CATHODE-RAY-TUBE A/D CONVERTERS

In block-diagram form, a CRT analog-to-digital converter is shown in Fig. 8.64. Its key element is cathode-ray tube (CRT) which consists of an electron gun, *1*, a modulator, *2*, deflection plates, *3* and *4*, an aperture plate or mask, *5*, and collector electrodes, *6*.

The voltage to be converted, V_{in}, is first amplified by an amplifier, *Ampl*, then applied to the vertical deflection plates, *4*. In this way, the beam is positioned at the quantizing level of the aperture plate corresponding to the input voltage. The horizontal

deflection plates, *3*, are energized with a sawtooth voltage supplied by a sweep generator, *SG*. This voltage causes the beam to scan the aperture plate in a horizontal direction. The aperture plate is placed in front of the collectors and has apertures arranged for binary coding as shown in the figure. The 1 in each digit position is represented by an aperture which allows the

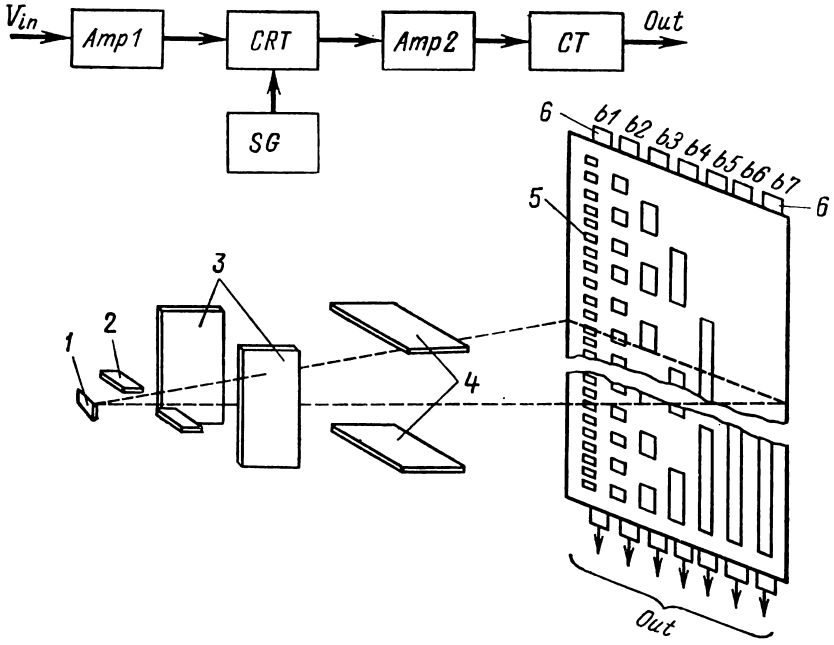

Fig. 8.64. Cathode-ray-tube convertor

beam to strike the respective collector, and the 0 in the same bit position by no aperture, so that the beam cannot reach the collector. The collector electrodes are applied to the vacuum side of the CRT envelope. They are arranged along the digit tracks, back of the aperture plate, and connected to the output lines.

As the beam sweeps across the aperture plate, it will strike the collectors only if it is directed at the respective apertures in the plate. As a result, a signal will appear on the respective output line and a binary 1 in the bit positions associated with that line. The duration and frequency of the sweep should be chosen such that the change in V_{in} during a sweep will remain within a prescribed error limit.

The quantizing step of a CRT converter is chosen according to the width of the aperture in the least significant position. The diameter of the sampling beam should be less than the width of the least significant bit aperture. Otherwise, false pulses might appear at the respective collector, and the result would be distorted.

An aperture plate using a straight binary code would cause considerable ambiguity errors. To avoid them, use should preferably be made of the cyclic (Gray) code or of the V-brush method.

The signals on the output lines of a CRT converter are rather weak and nonuniform. This is why they are first applied to amplifiers-shapers, *Amp2*, provided for each digit position. From these amplifiers, the code number is fed to a binary code translator, *CT*. The structure of the translator depends on the code used by the aperture plate, and the form of the signals it produces on the type of the device served by the converter.

CHAPTER 9

DIGITAL-TO-ANALOG CONVERSION

Digital-to-analog conversion is the reverse of analog-to-digital conversion. In analog-to-digital conversion, an analog quantity is first quantized, then presented by a sequence of binary numbers so that it can be handled by a digital computer.

In digital-to-analog conversion, digital data turned out by a computer are converted to an analog signal suitable for the subsequent use in process control or similar applications.

9.1. WEIGHTED-VOLTAGE D/A CONVERTERS

The basis of weighted-voltage D/A converters is the fact that any binary number $x_n x_{n-1} \ldots x_0$ may be expressed as a sum of powers of 2:

$$x_n x_{n-1} \ldots x_0 = \sum_{i=0}^{n} x_i 2^i = 2^n x_n + 2^{n-1} x_{n-1} + \ldots + 2^0 x_0$$

where x_i may be 1 or 0 and 2^i is the weight of a digit in a number.

Hence, if a binary number is to be converted to an analog quantity (such as a continuously varying voltage or current), each unit of the number may be assigned a voltage or current having a particular weight, and their sum then found.

The circuit of a simple three-stage weighted-voltage D/A converter is shown in Fig. 9.1. It has three voltage sources proportional to some power of 2, namely, $e_0 = 2^0 e$, $e_1 = 2^1 e$, and $e_2 = 2^2 e$, internal resistances r_0, r_1 and r_2, a load resistor R_L, polarized relays R_0, R_1 and R_2, biased to make the lower contact. When the coil of a polarized relay is energized with a *1* signal, the relay operates (moves upwards) and passes the respective weighted voltage on to output. As a result, a sum of weighted voltages forms at the output, proportional to the binary number being converted:

$$x_2 x_1 x_0 \rightarrow \sum_{i=0}^{2} x_i e_i$$

The output voltage of an n-digit converter is

$$\sum_{i=0}^{n} x_i e_i - i\left(\sum_{i=0}^{n} x_i r_i + R_L\right) = 0$$

whence

$$i = \sum_{i=0}^{n} x_i e_i \Big/ \left(\sum_{i=0}^{n} x_i r_i + R_L\right)$$

The voltage across the load resistor is

$$V_{out} = iR_L = \left[R_L \Big/ \left(\sum_{i=0}^{n} x_i r_i + R_L\right)\right] \sum_{i=0}^{n} x_i e_i$$

The ratio $R_L/\left(\sum_{i=0}^{n} x_i r_i + R_L\right)$ is called the scale factor and symbolized as K_m. Then

$$V_{out} = K_m \sum_{i=0}^{n} x_i e_i$$

or, on setting $e_i = 2^i e$, we finally get

$$V_{out} = K_m e \sum_{i=0}^{n} x_i 2^i$$

which shows that the output voltage, V_{out}, is proportional to the binary number being converted:

$$\sum_{i=0}^{n} x_i 2^i = X$$

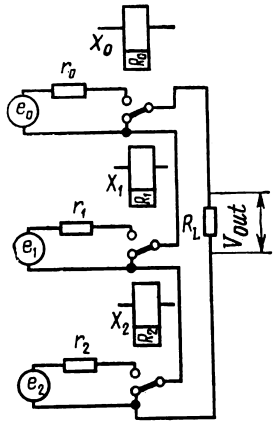

Fig. 9.1. Weighted-voltage summing network

The accuracy of conversion by the circuit described above depends on the constancy of the scale factor K_m and the stability of the weighted voltages.

The scale factor K_m is a function of the internal resistances of the voltage sources and of the load resistance; nor does it remain constant as the number is changed, decreasing with increasing count of ones in the number. To improve the stability of the scale factor, use should be made of voltage sources whose internal resistances are negligible in comparison with the load resistance, that is,

$$\sum_{i=0}^{n} x_i r_i \ll R_L$$

Then the scale factor may be taken constant and equal to unity, and

$$V_{out} = e \sum_{i=0}^{n} x_i 2^i$$

As before, the polarized relays may be replaced by electronic switches like flip-flops. For better accuracy, the switches in this circuit should have a low contact resistance in the energized condition.

When the supply source is on, the contact resistance $r_{c,i}$ of a switch acts in much the same manner as the internal resistance of the source, r_i. Therefore, instead of r_i in the equation for the output voltage we may use the sum $r_i + r_{c,i}$. Therefore, everything said about the effect of the internal resistance of a source applies to the contact resistance of a switch:

$$\sum_{i=0}^{n} (r_i + r_{c,i}) \ll R_L$$

9.2. WEIGHTED-RESISTOR D/A CONVERTERS

A weighted-resistor D/A converter uses a chain of resistors, R_i, whose values are chosen to be proportional to the weights of the digits in the binary number being converted, that is, to various powers of 2:

$$R_i = R/2^i$$

With such an arrangement,

$$\sum_{i=0}^{n} \frac{1}{R_i} = 2^0/R + 2^1/R + \ldots + 2^i/R + \ldots + 2^n/R = \frac{1}{R}(2^{n+1} - 1)$$

One of such circuits is shown in Fig. 9.2. It uses two voltage sources, E_1 and E_0 (in a special case, E_0 may be equal to zero), switches Sw_0, Sw_1, ..., Sw_i, ..., Sw_n, and weighted resistors R_0, R_1, ..., R_i, ..., R_n. Study of the diagram will show that the circuit functions as a digital potentiometer.

Initially, with 0's in all bit positions, the switches are in the down position, and all weighted resistors are connected to the source E_0. When a binary number $X = \sum_i 2^i x_i$ is applied to the converter, such that its jth digit positions hold 1's and its kth

positions hold 0's, the jth switches will connect the respective jth weighted resistors to the source E_1, and the kth switches will connect the respective kth weighted resistors to the source E_0. The equivalent circuit diagram applicable to this case is shown in Fig. 9.3.

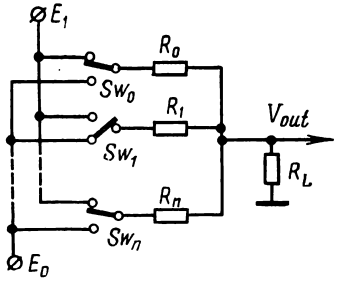

Fig. 9.2. Weighted-resistor circuit

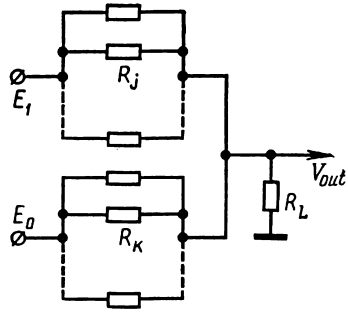

Fig. 9.3. Equivalent diagram of a weighted-resistor circuit

For the resistors R_j connected in parallel to the source E_1, the following equation may be written:

$$\sum_j 1/R_j = \sum_j 2^j/R = (1/R) \sum_j 2^j$$

Since, however,

$$\sum_j 2^j = \sum_i x_i 2^i = X$$

then

$$\sum_j 1/R_j = X/R$$

It may be concluded from the last equality that the sum $\sum_j (1/R_j)$ is a mapping of the number X being converted, on a scale such that $K_m = 1/R$.

Since

$$\sum_{i=0}^{n} (1/R_i) = \sum_j (1/R_j) + \sum_k (1/R_k)$$

then

$$\sum_k (1/R_k) = \sum_{i=0}^{n} (1/R_i) - \sum_j (1/R_j)$$

On substituting, we get

$$\sum_k (1/R_k) = (1/R)(2^{n+1} - 1 - X)$$

It can be shown that the output voltage V_{out} is proportional to the number X being converted.

On the basis of the superposition theorem, the output voltage may be visualized as consisting of two independent components:

$$V_{out} = V_{out\,1} + V_{out\,0}$$

where V_{out1} is due to the source E_1 (with $E_0 = 0$), and V_{out0} is due to the source E_0 (with $E_1 = 0$).

Let us determine these components.

(1) $E_0 = 0$

$$V_{out\,1} = E_1 \frac{1/\left[(1/R_L) + \sum_k (1/R_k)\right]}{1/\sum_j (1/R_j) + 1/\left[\left(1/R_L + \sum_k 1/R_k\right)\right]}$$

$$= E_1 \frac{1/\left[\left(1/R_L + \sum_k 1/R_k\right)\right]}{\left(1/R_L + \sum_j 1/R_j + \sum_k 1/R_k\right)/\left[\left(1/R_L + \sum_k 1/R_k\right)\sum_j 1/R_j\right]}$$

$$= E_1 \frac{\sum_j 1/R_j}{1/R_L + \sum_j 1/R_j + \sum_k 1/R_k}$$

Since in the last expression

$$\sum_j 1/R_j + \sum_k 1/R_k = \sum_{i=0}^{n} 1/R_i$$

then

$$V_{out\,1} = E_1 \frac{\sum_j 1/R_j}{1/R_L + \sum_{i=0}^{n} (1/R_i)}$$

(2) $E_1 = 0$

$$V_{out\,0} = E_0 \frac{1/\left(1/R_L + \sum_j 1/R_j\right)}{1/\sum_k 1/R_k + 1/\left(1/R_L + \sum_j 1/R_j\right)}$$

After similar manipulation, we get

$$V_{out\,0} = E_0 \frac{\sum_k 1/R_k}{1/R_L + \sum_{i=0}^{n} 1/R_i}$$

Thus, the total output voltage is

$$V_{out} = V_{out\,1} + V_{out\,0} = \frac{1}{1/R_L + \sum_{i=0}^{n} 1/R_i} \left(E_1 \sum_j 1/R_j + E_0 \sum_k 1/R_k \right)$$

Substituting for $\sum_{i=0}^{n} (1/R_i)$, $\sum_j (1/R_j)$ and $\sum_k (1/R_k)$ gives

$$V_{out} = \frac{1}{1/R_L + (2^{n+1} - 1)/R} \left[(E_1/R)\, X + (E_0/R)\, (2^{n+1} - 1 - X) \right]$$

$$= \frac{(E_1 - E_0)\, X + (2^{n+1} - 1)\, E_0}{R/R_L + 2^{n+1} - 1}$$

On setting

$$\frac{1}{R/R_L + 2^{n+1} - 1} = K_{m1}$$

and

$$\frac{2^{n+1} - 1}{R/R_L + 2^{n+1} - 1} = K_{m2}$$

the equation for V_{out} may be re-written as

$$V_{out} = K_{m1} (E_1 - E_0)\, X + K_{m2} E_0$$

This equation shows that V_{out} is a linear function of the number being converted.

In the special case of $E_0 = 0$,

$$V_{out} = K_{m1} E_1 X \tag{9.1}$$

The overall error, ΔV_{out}, in the output voltage may be found, if the primary errors are known, by evaluating the total differential as the sum of the partial differentials.

Consider the accuracy of the converter for $E_0 = 0$.

$$\Delta V_{out} = (\partial V_{out}/\partial E_1)\, \Delta E_1 + (\partial V_{out}/\partial R_L)\, \Delta R_L + (\partial V_{out}/\partial R)\, \Delta R$$

where ΔE_1 is the error due to instability of the supply voltage E_1, ΔR_L and ΔR are the errors due to instabilities in and spread between the resistors.

Consider these errors separately.

The error due to instabilities in E_1 may be found by differentiating Eq. (9.1) with respect to E_1:

$$\Delta V_{out\ E_1} = \frac{XR_L}{R + (2^{n+1} - 1) R_L} \Delta E_1 \tag{9.2}$$

Hence, the fractional error is

$$\delta V_{out} = \Delta V_{out\ E} / V_{out\ \max} \tag{9.3}$$

The output voltage, Eq. (9.1), will be a maximum when the number being decoded takes on a maximum value, that is, at $X = 2^{n+1} - 1$:

$$V_{out\ \max} = \frac{(2^{n+1} - 1) R_L E_1}{R + (2^{n+1} - 1) R_L} \tag{9.4}$$

Substituting (9.2) and (9.4) in (9.3) yields

$$\delta V_{out} = (\Delta E_1 / E_1) \frac{X}{(2^{n+1} - 1)} \tag{9.5}$$

This expression shows that the fractional error increases with increasing number being decoded; its maximum value is

$$\delta V_{out\ \max} = \Delta E_1 / E_1 \tag{9.6}$$

The error due to instabilities in the load resistance may be found in a similar way:

$$\Delta V_{out\ R_L} = (\partial V_{out} / \partial R_L) \Delta R_L = \frac{XRE_1}{[R + (2^{n+1} - 1) R_L]^2} \Delta R_L \tag{9.7}$$

The fractional error in this case is

$$\delta V_{out\ R_L} = \Delta V_{out\ R_L} / V_{out\ \max}$$

$$= \frac{R}{[R + (2^{n+1} - 1) R_L]} \frac{X}{(2^{n+1} - 1)} \frac{\Delta R_L}{R_L} \tag{9.8}$$

Accordingly, the maximum fractional error is

$$\delta V_{out\ R_L\ \max} = \frac{R}{[R + (2^{n+1} - 1) R_L]} \frac{\Delta R_L}{R_L} \tag{9.9}$$

The error in the output voltage due to the fact that the actual weighted resistors differ from the design values is found as the sum of departures of the individual resistors.

The overall error including that due to all weighted resistors is

$$\Delta V_{out\ R_w} = E_1 \frac{(n+1)\, R_L\, \Delta R X}{[R \pm \Delta R + (2^{n+1} - 1)\, R_L]^2} \tag{9.10}$$

It is assumed that the departure ΔR is the same in all digit positions and that account is taken of only the respective weights.

The fractonal error due to the weighted resistors is

$$\delta V_{out\ R_w} = \frac{(n+1)\, X}{2^{n+1} - 1} \frac{\Delta R}{[R \mp \Delta R\, (2^{n+1} - 1)\, R_L]} \tag{9.11}$$

The error in output voltage due to the contact resistance of switches manifests itself in much the same way as that due to the weighted resistors and may be found by Eqs. (9.10) and (9.11). However, in referring the contact resistance of each switch to ΔR_L, account must be taken of their respective weights.

The scale factor in Eq. (9.1) is a function of the digit range of the binary number being converted. As the digit range increases, the scale factor decreases, and so does the amplitude of the output voltage. Also, the scale factor is a function of resistance, and this impairs the accuracy of output voltage. These limitations may be removed by using weighted resistors in combination with a d.c. amplifier.

9.3. D/A CONVERTERS USING A D.C. AMPLIFIER

A d.c. amplifier with a large amount of negative feedback is widely used as a computing element in analog computers, notably as a summing amplifier. Fitted with weighted resistors at its input, a summing amplifier may well serve to convert numbers to voltage. As a proof, we shall examine the circuit of Fig. 9.4.

If the input resistance of the d.c. amplifier is sufficiently high, the feedback current may be assumed to be

$$i_{fb} = i_0 + i_1 + \ldots + i_n \tag{9.12}$$

By Kirchhoff's voltage law,

$$e_0 = R_0 i_0 + R_{fb} i_{fb} + V_{out}$$
$$e_1 = R_1 i_1 + R_{fb} i_{fb} + V_{out}$$
$$\cdots\cdots\cdots\cdots$$
$$e_n = R_n i_n + R_{fb} i_{fb} + V_{out}$$

and

$$-V_{out} = kV_b = k(R_{fb} i_{fb} + V_{out})$$

where V_b is the input voltage of the amplifier.

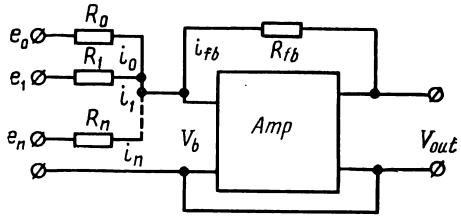

Fig. 9.4. Converter using a summing amplifier with weighted resistors at input

Similarly, by Kirchhoff's current law,

$$i_0 = (e_0 - V_{out} - R_{fb} i_{fb})/R_0$$
$$i_1 = (e_1 - V_{out} - R_{fb} i_{fb})/R_1$$
$$\cdots\cdots\cdots\cdots$$
$$i_n = (e_n - V_{out} - R_{fb} i_{fb})/R_n$$

and

$$i_{fb} = -\frac{1+k}{R_{fb}k} V_{out}$$

Substituting the above expressions in Eq. (9.12) gives

$$-\frac{1+k}{R_{fb}k} V_{out} = \frac{e_0 - V_{out} - R_{fb} i_{fb}}{R_0}$$
$$+ \frac{e_1 - V_{out} - R_{fb} i_{fb}}{R_1} + \ldots + \frac{e_n - V_{out} - R_{fb} i_{fb}}{R_L}$$

whence

$$-\frac{1+k}{k} V_{out} = R_{fb} [e_0/R_0 + e_1/R_1 + \ldots + e_n/R_L$$
$$-(V_{out} + R_{fb} i_{fb})(1/R_0 + 1/R_1 + \ldots + 1/R_n)]$$

Recalling that

$$V_{out} + R_{fb} i_{fb} = - V_{out}/k$$

we may write

$$-\frac{1-k}{k} V_{out} = R_{fb} (e_0/R_0 + e_1/R_1 + \ldots + e_n/R_L)$$
$$+ (V_{out}/k)(1/R_0 + 1/R_1 + \ldots + 1/R_L) \qquad (9.13)$$

For a d.c. amplifier with a sufficiently high gain ($k \gg 1$), we may write to a fairly good degree of accuracy that

$$(1 + k)/k = 1/k + 1 = 1$$

and

$$(V_{out}/k)(1/R_0 + 1/R_1 + \ldots + 1/R_L) = 0$$

Accordingly, Eq. (9.13) may be re-written as

$$V_{out} = - R_{fb} (e_0/R_0 + e_1/R_1 + \ldots + e_n/R_L) \qquad (9.14)$$

Since the amplifier has a high input resistance and has no effect on the input circuit, the load resistance is not included in Eq. (9.14) and its variation has no effect on accuracy.

Equation (9.14) shows that the output voltage of the amplifier is governed by three quantities, namely the feedback resistance R_{fb}, the input-circuit resistances R_0, R_1, ..., R_n, and the supply voltages.

Accordingly, the circuit may be employed in any one of three variants.

Variant 1. R_{fb} = constant, $R_0 = R_1 = \ldots = R_n = R$ = constant, $e_0 = 2^0 e$, $e_1 = 2^1 e$, ..., $e^n = 2^n e$, that is, the amplitudes of input voltages have weights proportional to powers of 2. The presence of a weighted voltage $e_i = 2^i e$ is recognized as a binary 1, and its absence, as a binary 0.

For this circuit, Eq. (9.14) can be written as

$$V_{out} = (- R_{fb}/R) \sum_{i=0}^{n} e_i = - e (R_{fb}/R) \sum_{i=0}^{n} 2^i x_i$$
$$x_i = 0 \text{ or } x_i = 1$$

Since, however, $\sum_{i=0}^{n} 2^i x_i$ is the number being converted, then

$$V_{out} = - e (R_{fb}/R) X$$

Variant 2. R_{fb} = constant, $e_0 = e_1 = \ldots = e_n$ = constant, $R_0 = R/2^0$, $R_1 = R/2^1$, ..., $R_n = R/2^n$, that is, the resistances at the input to the amplifier

have weights proportional to powers of 2. Connection of the ith resistor at the input is recognized as a 1 in the ith digit position of the number being converted, and disconnection of the same resistor, as a 0:

$$V_{out} = -R_{fb}e \sum_{i=0}^{n} (1/R_i) = -R_{fb}e\,(1/R) \sum_{i=0}^{n} 2^i x_i$$

Variant 3. The circuit implementing this variant is shown in Fig. 9.5. A constant voltage, e, is applied to the amplifier input via a resistance R (R = con-

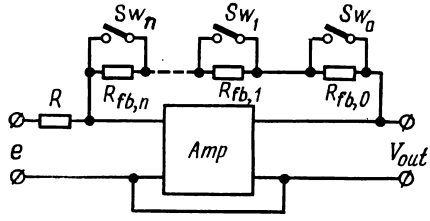

Fig. 9.5. Converter using an amplifier with weighted resistors in the feedback path

stant, e = constant). The feedback resistance, R_{fb}, is divided into a sum of resistances with assigned weights proportional to some powers of 2:

$$R_{fb} = 2^n R'_{fb} + \ldots + 2^1 R'_{fb} + 2^0 R'_{fb} = R_{fb,\,n} + \ldots + R_{fb,\,1} + R_{fb,\,0}$$

Each weighted resistor $2^i R'_{fb}$ is switched into or out of the feedback path by a switch, according as the ith digit position of the binary number being converted contains a 1 or a 0. The switches are controlled by the digit-position signals. If a digit position contains a 1, the respective switch will be open; if the same digit position contains a 0, the same switch will be closed. With this arrangement, the resultant resistance of the feedback path is proportional to the number X being converted.

Hence,

$$V_{out} = -(e/R) \sum_{i=0}^{n} 2^i R'_{fb} x_i$$

$$= -e\left(R'_{fb}/R\right) \sum_{i=0}^{n} 2^i x_i = -e\left(R'_{fb}/R\right) X$$

In all the three cases, the conversion accuracy depends on the primary errors due to instabilities in the supply voltage e, instabilities in, and spread of R and R_{fb}.

The overall absolute error in output voltage can be found as before, by evaluating the total differential of the function as the sum of partial differentials.

In evaluating the partial differentials for each variant, it is important to take account of the weights:

for variant 1:

$$\Delta V_{out} = \sum_{i=0}^{n} (\partial V_{out}/\partial e_i)\,\Delta e_i + (\partial V_{out}/\partial R)\,\Delta R + (\partial V_{out}/\partial R_{fb})\,\Delta R_{fb}$$

for variant 2:

$$\Delta V_{out} = (\partial V_{out}/\partial e)\,\Delta e + \sum (\partial V_{out}/\partial R_i)\,\Delta R_i + (\partial V_{out}/\partial R_{fb})\,\Delta R_{fb}$$

for variant 3:

$$\Delta V_{out} = (\partial V_{out}/\partial e)\,\Delta e + (\partial V_{out}/\partial R)\,\Delta R + \sum_{i=0}^{n} \left(\partial V_{out}/\partial R'_{fb,\,i}\right)\Delta R'_{fb,\,t}$$

From a comparison of the three variants, it may be concluded that variants 2 and 3 are simpler to instrument as they need only passive elements (resistors). From the view-point of precision, however, variant 1 is more attractive. While with variants 2 and 3, the precision does not exceed five or six binary digits, variant 1 offers a greater precision through a proper choice of weighted voltage sources.

9.4. CURRENT-SUMMATION D/A CONVERTERS

As before, the key elements of a D/A converter based on current summation are weighted resistors. In contrast, however, the weighted resistor of each digit position is energized from a current source of its own. Consider the design and operation of a current-summation converter, taking a five-unit circuit shown in Fig. 9.6 as an example. It consists of current stabilizers, CS_i, weighted resistors, $R_i = 2^i R$, and switches, Sw_i. The switches are positioned according to the values in the digit positions of the binary number being converted. A 1 in a digit position causes the right-hand chain of resistors to be brought in circuit; a 0 switches in the left-hand chain.

The output voltage, V_{out}, is determined as the difference in potential between points A and B:

$$V_{out} = V_A - V_B$$

The potential at point A is determined by the count of zeros in the binary number being converted.

If all digit positions of the number are zeros, $X_0 = 00000$, all switches will reside on the left-hand contacts, and the potential at point A will be

$$V_A = E - (I_0 R + I_1 2R + I_2 4R + I_3 8R + I_4 16R)$$

If all stabilizers furnish the same current I:

$$I = I_0 = I_1 = I_2 = I_5 = I_6$$

then the expression for the potential at point A will take the form

$$V_A = E - IR \sum_{i=0}^{4} 2^i = E - 31IR$$

The potential at point B will then be equal to the supply voltage, $V_B = E$.

For non-zero numbers, that is, those containing various combinations of ones and zeros in their digit positions, the setting of the switches will change accordingly. This can be taken care of by introducing, as usual, the factor $x_i = 0$ or 1, which represents the value of the ith digit position in the binary number $X = x_4x_3x_2x_1x_0$.

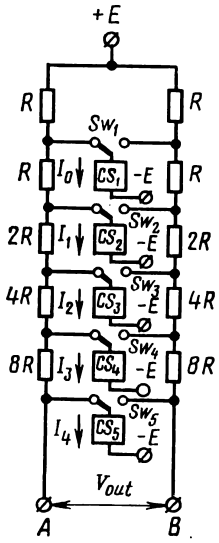

Fig. 9.6. Current-summation circuit

In such a case, the potential at point A will be incremented by an amount proportional to the binary number

$$V_A = E - 31IR + (2^0x_0IR + 2^1x_1IR + 2^2x_2IR + 2^3x_3IR + 2^4x_4IR) = E - 31IR + IR \sum_{i=0}^{4} 2^ix_i = E - 31IR + IRX \quad (9.15)$$

Accordingly, the potential at point B will be decremented by an amount proportional to the same binary number:

$$V_B = E - IRX$$

Hence, the output voltage will be

$$V_{out} = V_A - V_B = 2XIR - 31IR$$

Designating $2IR = K_{m1}$ and $31IR = K_{m2}$, we get

$$V_{out} = K_{m1}X - K_{m2}$$

This equation shows that the output voltage is a linear function of the binary number X being converted.

In this expression the values of the scale factors K_{m1} and K_{m2} only hold for the five-unit converter examined.

In the general case, for the conversion of an n-digit binary number Eqs. (9.15) and (9.16) take the form

$$V_A = E - (2^n - 1)\,IR + IR \sum_{i=0}^{n} 2^i x_i$$

$$V_B = E - IR \sum_{i=0}^{n} 2^i x_i$$

Hence, the output voltage is

$$V_{out} = V_A - V_B = 2IR \sum_{i=0}^{n} 2^i x_i - (2^n - 1)\,IR$$

Noting that $\sum_{i=0}^{n} 2^i x_i = X$ and designating $2IR = K_{m1}$ and $(2^{n-1} - 1)\,IR = K_{m2}$, we get

$$V_{out} = K_{m1} X - K_{m2}$$

The accuracy of current-summation converters, as follows from the last expression, depends on the equality of all supply voltages, departure of the weighted resistors from their nominal values. and instabilities in currents and resistances.

The overall error in output voltage is

$$\Delta V_{out} = \sum_{i=0}^{n} (\partial V_{out}/\partial R_i)\,\Delta R_i + (\partial V_{out}/\partial I_i)\,\Delta I_i$$

where ΔR_i is the error in the weighted resistor, and ΔI_i is the error in the supply current.

9.5. TWO-RESISTANCE-VALUE CONVERTERS

In the previous cases, use was made of a set of elements whose weights were chosen to be proportional to the digit positions of the binary number being converted. However, it is possible, through an appropriate circuit arrangement, to arrive at the same result by using supply sources delivering the same current or voltage and only a set of two resistance values. The number of supply sources should be equal to that of the digit positions in the number being converted, and the resistors should form a divider such that the output signal due to each source will have a weight proportional to the respective bit.

The circuit of a converter using current sources is shown in Fig. 9.7. It consists of identical current sources, I_i, resistors of two values, R and $2R$, and switches, Sw_i. A switch will be closed if the associated digit position holds a 1; otherwise, this switch will remain open.

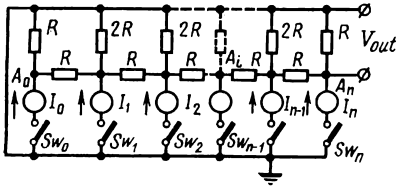

Fig. 9.7. Converter using current sources

The grid-forming resistors are connected to the current sources so that the load on any inner source (from $i = 1$ to $i = n - 1$) will be $2R$ on the left, right and top. As a consequence, the total load R_i on each inner source will be

$$1/R_i = 1/2R + 1/2R + 1/2R = 3/2R$$

whence

$$R_i = 2R/3$$

The load on the inner current sources can readily be determined by inspection of the circuit in Fig. 9.8. The load on the leftmost

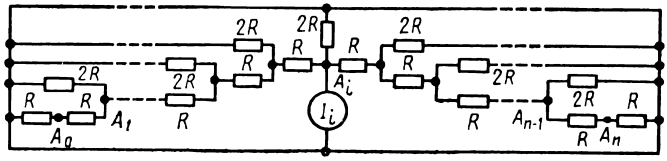

Fig. 9.8. Equivalent circuit diagram of inner current source load resistances

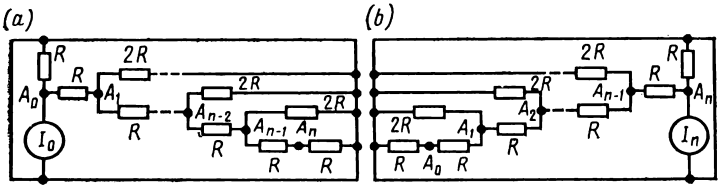

Fig. 9.9. Equivalent circuit diagram of outer current source load resistances

current source representing the zero digit position, x_0, can be determined by inspection of the circuit shown in Fig. 9.9*a*:

$$1/R_0 = 1/R + 1/2R$$

or

$$R_0 = 2R/3$$

The load on the rightmost current source representing the most significant digit of the binary number, x_n, can be determined from the circuit of Fig. 9.9*b*:

$$1/R_n = 1/R + 1/2R = 3/2R$$

or

$$R_n = 2R/3$$

It may be concluded that all stages are loaded by equal resistances. The voltage drop E across this load in each stage due to its own source is

$$E_i = 2RI_i/3$$

If the output voltage is sampled from the load of the last stage, as is done in the circuit of Fig. 9.7, the voltage drop E_n due to the current source I_n will fully appear at output. The voltage drop E_{n-1} in the preceding stage is halved as it is passed on to output via a divider, $R + R$. The voltage drop of the $(n-2)$nd stage is quartered as it is passed on to output. Thus, the voltage drop E_i in the ith stage is divided by $2^{(n-i)}$ as it is passed on to output. Hence, the output voltage due to the current sources of all stages is

$$E_{out} = E_0/2^n + E_1/2^{n-1} + \ldots + E_i/2^{n-i} + \ldots + E_n/2^0$$

If the binary number X may be written as

$$X = x_n 2^n + x_{n-1}2^{n-1} + \ldots + x_1 2^1 + x_0 2^0$$

then assigning a 1 or a 0 to x_i according as the respective current source is brought in or out of circuit, the equation for the output voltage may be written as

$$\begin{aligned} E_{out} &= (E_0/2^n)\, x_0 + (E_1/2^{n-1})\, x_1 + \ldots \\ &\quad + (E_i/2^{n-i})\, x_i + (E_n/2^0)\, x_n \\ &= (2/3)\, RI\, (1/2^n)\, (x_0 2^0 + x_1 2^1 + \ldots \\ &\quad + x_i 2^i + \ldots + x_n 2^n) = (2/3)\, RI\, (1/2^n)\, X; \\ &(I_0 = I_1 = \ldots = I_n = I) \end{aligned}$$

This equation shows that the output voltage is a linear function of the binary number X being converted. The accuracy of conversion depends on the stability of the scale factor

$$K_m = (2/3)\, RI\, (1/2^n)$$

The stability of the latter depends in turn on instabilities in and equality of supply currents, instabilities and spread in the values of R and $2R$.

Resistor grids of only two resistance values may be used in conjunction with voltage sources as well. An example of such an arrangement is shown in Fig. 9.10. In this circuit, the load resistance on each source is $3R/2$.

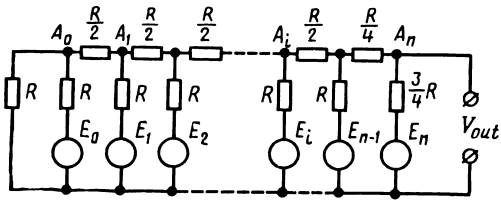

Fig. 9.10. Converter using voltage sources

The voltage, V_k, between the point k and ground, due solely to source E_k is

$$V_k = E_k \frac{R/2}{R + R/2} = E_k/3$$

This voltage is halved at each inner node to the right of the point k.

The voltage, V_{n-1}, between the point $n-1$ and ground, due solely to source E_k is

$$V_{n-1} = \frac{V_k}{2^{n-1-k}} = (E_k/3)\left(1/2^{n-1-k}\right)$$

If V_{n-1} is known, it is an easy matter to determine the output voltage due to E_k, that is

$$E_{out\ Ek} = \frac{V_{n-1}(3R/4)}{3R/4 + R/4} = V_{n-1}(3/4) = (E_k/4)\left(1/2^{n-1-k}\right)$$

The output voltage due to the nth source, E_n, is

$$E_{out\ En} = \frac{E_n 3R/4}{3R/4 + 3R/4} = E_n/2$$

The output voltage due to all sources may be found by adding together the effects due to each source, that is, from $k = 0$ to $k = n$:

$$E_{out} = (1/4)\left(E_0/2^{n-1} + E_1/2^{n-2} + \ldots + E_n/2^1\right)$$

$$= \left(1/2^{n+1}\right)\left(E_0 + 2^1 E_1 + \ldots + 2^n E_n\right) = \left(1/2^{n+1}\right)\sum_{k=0}^{n} E_k 2^k$$

Using the above equation, we can readily determine the output voltage proportional to the binary number being converted, $N = x_n x_{n-1} \ldots x_0$.

If all voltage sources E_k supply the same voltage E and are brought in circuit according as the respective digit position contains a 1 or a 0, the output voltage will be proportional to the number being converted:

$$E_{out} = (E/2^{n+1})(x_n 2^n + x_{n-1}2^{n-1} + \ldots + x_0)$$

9.6. SERIAL BINARY NUMBER-TO-VOLTAGE CONVERSION

So far we dealt with the conversion of binary numbers to voltage on the assumption that the numbers came in parallel form. If the number to be converted comes in a serial code, that is, bit by bit, it will have to be converted to a parallel form if the circuits discussed previously are to be of any use. This purpose may be served by, say, a serial-parallel register.

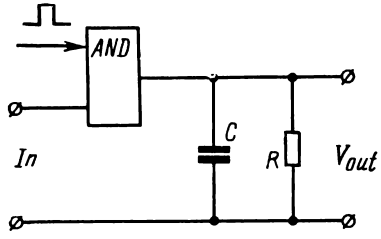

Fig. 9.11. Storage-capacitor converter

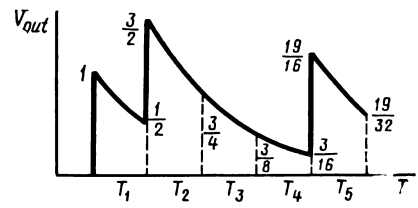

Fig. 9.12. Plot of output voltages

On the other hand, a serial number can be converted to an analog quantity without translating it first to a parallel code. This involves the use of a storage capacitor in order to store the weighted voltages proportional to the respective digit positions of the number being converted. The schematic diagram of a storage-capacitor D/A converter is shown in Fig. 9.11.

The binary number to be converted is read serially, the least significant digit first, into the *AND* gate. The other input of the *AND* gate accepts square pulses calibrated in amplitude and duration, in step with the number pulses. The *AND* circuit gates out the calibrated pulses if the respective digit position holds a 1, and does not if the same position is occupied by a 0. Thus the output combination is again a serial binary number, but coded with calibrated pulses. Then the pulses go to a capacitor, *C*, and charge it only at a specific clock time if the value of the particular digit is 1.

For correct operation of the circuit, it is essential that the clock time T between discharges be held constant and that the RC-network have a time constant $\tau = RC$ such that the capacitor will discharge to half its value during the time T. Consider the operation of such a converter, using the conversion of $X = 10011$ as an example (Fig. 9.12).

The least significant digit charges the capacitor to a voltage equal to 1. By the time the next higher significant digit comes, that is during the clock period T, the capacitor discharges to one-half of its value. The next higher digit increments the charge on the capacitor by 1 again, so that it is 3/2 now. During the second clock period the capacitor discharges to half its value, that is, to 3/4, etc. After five clock periods, the total charge on the capacitor will be

$$V_{out}\{\{[(V/2 + 1 \times V)/2 + 0 \times V]/2 + 0 \times V\} + 1 \times V\}/2 = 19V/2^5 = VX/2^5$$

On introducing the scale factor $K_m = V/2^5$, we get

$$V_{out} = K_m X$$

This expression shows that the output voltage after five clock periods is proportional to X^5. This equation may be extended to the more general case involving an n-digit number. It is important to sample the output voltage at a precisely specific instant (immediately after the last clock period). Otherwise, the result would be in error. The accuracy of this circuit is governed by the constancy of R and C and by the calibration of timing or clock pulses.

9.7. NUMBER-TO-SHAFT POSITION CONVERSION

Feedback technique. A number-to-shaft position converter is shown in block-diagram form in Fig. 9.13. Here the number A which has been generated during computation represents the required shaft position. The comparator samples both the number A and the number B from the A/D shaft-position converter, representing the actual shaft position, and, on the basis of comparison of the two digital numbers, generates a discrete error signal. This error signal is then converted into an electric signal which is boosted in voltage and power and drives the servo motor towards the correct shaft position, that is, so as to minimize the error (or difference) signal, $A - B$.

In this arrangement, the A/D shaft position converter may be any type examined earlier. Leaving out the amplifier and servo motor which are identical with those used in the systems to be discussed later, consider the operation of the comparator and

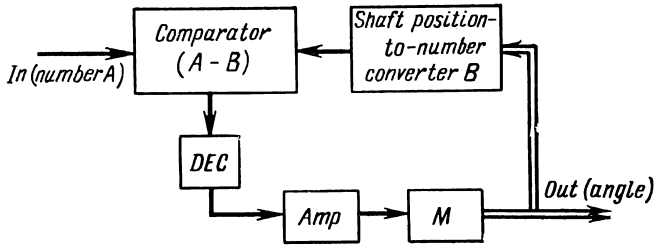

Fig. 9.13. Block diagram of a number-to-shaft position converter

decoder. The comparator compares the two digital numbers A and B and, on the basis of this comparison, selects the direction in which the error will be minimized in the shortest time.

Four cases may arise in the operation of the comparator. They are illustrated in Fig. 9.14*a*, *b*, *c* and *d*, respectively.

In the first case, the difference between the required and actual shaft positions is less than 180° and positive. To minimize the difference, or error, the servo shaft B should rotate counter-clockwise.

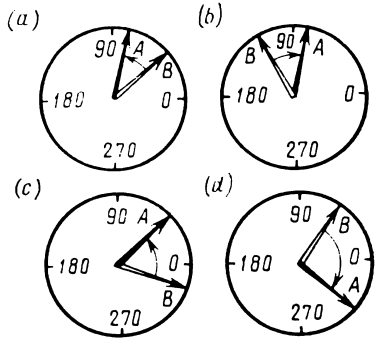

Fig. 9.14. Four cases of error

In the second case, the difference is again less than 180°, but negative. Now the servo shaft should be rotated clockwise in order to achieve the correct position.

In the third case, the angular position of B is greater than that of A, and the difference $A - B$ is negative. Therefore, the servo shaft should be rotated counter-clockwise in order to minimize the error.

In the fourth case, the difference $A - B$ exceeds 180° and is positive. To minimize the error, the servo shaft should be rotated clockwise.

Thus, in order to minimize the error, the servo shaft should be rotated counter-clockwise in the first and third cases, and clockwise in the second and fourth cases.

These requirements can readily be met if the numbers A and B are read into the comparator in straight binary form, and their difference is generated in 1's complement form. This is because the difference in the first and third cases will always be expressed as

$$A - B = x_{n-1}x_{n-2} \ \ldots \ x_0 = 0 \cdot x_{n-2} \ \ldots \ x_0$$

Since the difference in the first case is less than half a shaft revolution and positive, the digit in the most significant position has the value 0, while in the third case the difference is greater than a half-revolution but negative, and so the digit in the most significant position will again have the value 0.

In the remaining two cases, the difference will be expressed as

$$A - B = 1 \cdot x_{n-2} \ \ldots \ x_0$$

that is, the most significant digit will have the value 1. Then, in the second case where the difference is less than a half-revolution and negative, the number in 1's complement form will have the value 1 in the most significant bit; in the fourth case, where the difference is greater than a half-revolution and positive, the most significant bit will again have the value 1.

Presentation of the difference in 1's complement form makes it possible to determine the direction in which the error is to be minimized by the digit in its most significant position. A 0 in the most significant position of the difference is an indication that the shaft should be rotated counter-clockwise in order to minimize the error, while a 1 in the same position is an indication that this should be done in the clockwise direction. The respective signal to control the servo motor is supplied by the decoder and amplifier. A likely circuit of the decoder is shown in Fig. 9.15.

The decoder consists of a flip-flop register comprising $2n - 1$ flip-flops, *FF*, *AND* gates *1* through *6*, and *NAND* gates *1* through *5*.

The decoder is synchronized by clock pulses t_1 and t_2. The numbers A and B are read into the comparator in synchronism and serially, with the least significant digit first. From the comparator, the difference is read into the decoder likewise serially, but in 1's complement form, with its least significant digit first.

Just as the most significant bit in the difference $A - B$ is moved from the comparator through the *AND1* gate, the *NAND1* gate and flip-flop FF_n, the control unit, *CU*, sends out a pulse to the other input of the *AND1* gate and the *NAND1* gate. If the most significant bit of the difference has the value 1, a pulse will appear at the output of the *AND1* gate and there will be no pulse at the output of the *NAND1*. If, on the other hand, the most

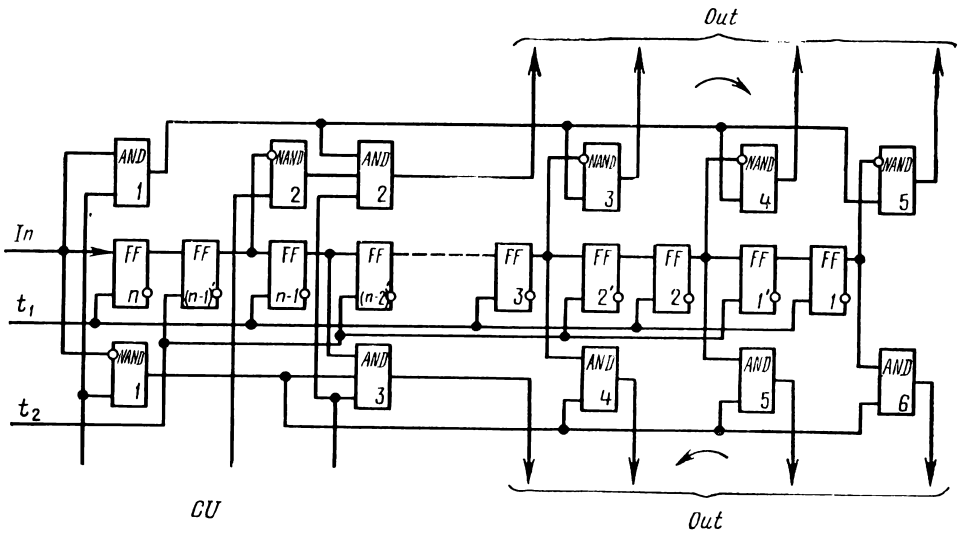

Fig. 9.15. Difference (error) signal decoder using magnetic core-transistor cells

significant bit of the difference has the value 0, there will be no pulse at the output of the *AND1* gate and a pulse will appear at the output of the *NAND1*. As is seen, either the upper arm (in the diagram) or the lower arm of the circuit generates an output pulse, according as the most significant bit has the value 1 or 0. In this way, a control signal is applied to the amplifier input in order to drive the servo shaft clockwise in the former case and anti-clockwise in the latter.

If the most significant bit of the difference has the value 0, the output pulse of the *NAND1* gate goes to the inputs of the *AND* gates *3* through *6*. At the same time, the other inputs of these gates accept the values contained in the $(n-1)$st, third, second and first digit positions of the difference from the outputs of

flip-flops FF_{n-1}, FF_3, FF_2 and FF_1, respectively. If a given position contains a 1 and the most significant bit is a 0, pulses will appear at the outputs of the *AND* gates, and the servo motor will be fed a control signal which will drive it counter-clockwise.

Consider operation of the upper arm in the diagram. The pulse generated by the *AND1* gate goes to the input of the *AND2* gate and to the *NAND* gates *3*, *4* and *5*. The other inputs of the *NAND* gates accept the signals from the register flip-flops FF_3, FF_2 and FF_1, respectively. If the given digit position contains a 0 and the most significant bit has the value 1, a pulse will appear at the output of the respective *NAND* gate to drive the motor clockwise.

The difference $A - B$ is minimized proportionally within the three least significant bits. Should a greater difference result, that is, should 1's appear in the digit positions higher than the third, the servo motor will be caused to minimize this error at the highest possible rate to that lying within the three least significant digits. This is accomplished as follows.

As the difference is read bit by bit into the delay lines, each bit passes through flip-flop FF_{n-1} to reach one of the inputs of the *AND3* gate. After the first three bits, the control unit generates a pulse to reset the *AND3* gate. This reset pulse cannot get through because the other input of the *AND3* gate receives no enabling pulse from the *NAND1* gate. When the next higher bits are read into the *AND3* gate, no reset pulses are applied. Should at least one of the higher bits contain a 1, this will be written into and stored by the *AND3* gate until a reset pulse comes. When the last but one bit of the difference signal comes from register flip-flop FF_{n-1} to the first input of the *AND3* gate and a "run counter-clockwise" signal from the *NAND1* gate to the other input, the control unit applies a "reset" pulse to the *AND3* gate, and the latter gates out an output signal.

When the difference exceeds the three least significant bits a "run clockwise" signal is generated by the *AND2* gate and the *NAND2* gate because the succeeding bits have to be inverted. To this end, it is arranged so that the inhibit input of the *NAND2* gate accepts the next higher bits of the difference from flip-flop $FF_{(n-1)}$, and the main input of the same *NAND* gate is fed pulses from the control unit. Should any of these higher bits have a value 0, the *NAND2* gate will produce an output pulse which will be written into the *AND2* gate. If the most significant bit of the difference is a 1, the other input of the *AND2* gate will likewise

be fed a pulse. Now, arrival of a "reset" pulse from the control unit will cause the gate to produce an output signal.

The generated signals should be added together according to the weight of the 1 in a particular digit position. This can be achieved by using, say, a magnetic stage having four pairs of input windings arranged so that their turns are proportional, that is

$$W_4 = 2W_3 = 4W_2 = 8W_1$$

where W_1 is the number of turns in the winding energized with the signal from the first digit position, W_2 is the number of turns in the winding energized from the second digit position, etc.

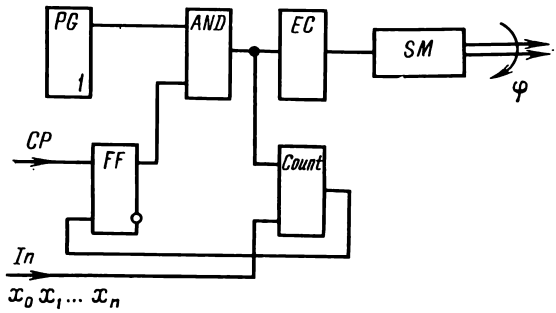

Fig. 9.16. Block diagram of a step-motor D/A converter

The output leads of the lower arm (in the diagram) are connected to the windings in anti-phase in comparison with those from the upper arm, in order that the servo motor can be reversed.

The input signal of the magnetic stage is proportional to the difference (error) signal in magnitude and has the sign appropriate to that of the error. This signal is amplified and applied to the servo motor in order to minimize the error.

Step-motor techniques. Step motors are widely used in number-to-position converters. This popularity is explained by the fact that the circuitry is simple because there is no feedback or intermediate number-to-analog voltage conversion involved, and also by the high dynamic performance of step motors.

A typical number-to-shaft position converter utilizing a step motor (Fig. 9.16) also incorporates a device to convert the digital number into a sequence of pulses (unitary code). Operation of

this device reduces to the following. Prior to a conversion cycle, the binary number $x_n x_{n-1} \ldots x_0$ to be converted is read into a subtract (or down) counter, *Count*. Then a control pulse, *CP*, causes the flip-flop, *FF*, to change state and to enable the *AND* gate. The latter gates out duration-calibrated pulses supplied by a pulse generator, *PG*, to the subtract counter and, via an electronic commutator, *EC*, to the step motor, *SM*. Each pulse read into the counter decrements the number it holds by one. As soon as the number of pulses written into the counter becomes equal to the number stored in it (the counter is said to be nulled), a pulse is generated, which resets the flip-flop. As a result, the *AND* gate is disabled, and no pulses can any longer pass through it. The number of pulses read into the counter is equal to that applied to the step motor.

Therefore,

$$\varphi = \Delta\varphi N$$

where φ = total angle of rotation of the step motor
$\Delta\varphi$ = angle of rotation of the step motor per driving pulse
N = number of pulses applied to the step motor defined as

$$N = k_1 X = k_1 \sum_{i=0}^{n} x_i 2^i$$

where X = number being converted
k = proportionality factor

If the range of binary number to be converted is known in advance and the largest number X_{max} corresponds to the number N_{max} of unitary-code pulses, then the maximum angle of rotation of the step motor, $\varphi_{max} = \Delta\varphi N_{max}$, may exceed 360°. In such cases, the output shaft of the converter should be coupled to the step motor via a reduction gear train so that the maximum angle of rotation of the step motor will not exceed 360°.

In cases where the step-motor number-to-position converter operates on sequences of numbers coming from a control computer, the counter should store the difference between the current and the previous numbers. Otherwise, the result might be in error.

The electronic commutator receives the unitary code and distributes the signals among the control windings. The circuit configuration of the electronic commutator depends on the number of phases in the step motor and their sequence.

As an example, consider the operation of the six-bit commutator shown in Fig. 9.17. It consists of flip-flops *FF1*, *FF2* and *FF3*, two *AND* gates, two *NAND* gates, and three *OR* gates. One of the inputs drives the step motor in one sense, and the other, in the opposite.

Suppose a sequence of pulses arrives at *In1*. The circuit is caused to operate by positive pulses. Initially, the non-inverting output of *FF1* is low and the inverting output of *FF2* and that

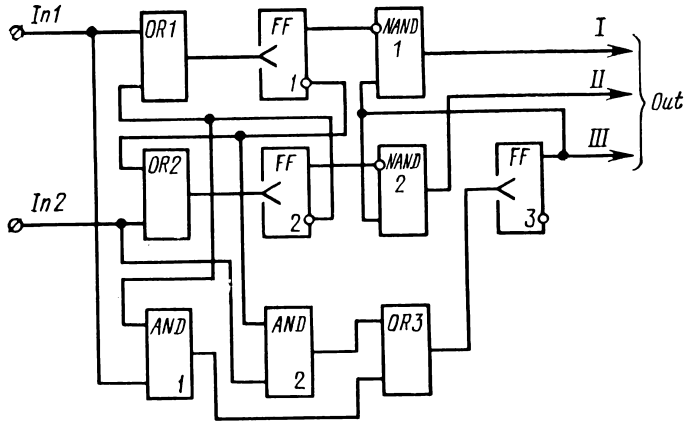

Fig. 9.17. Six-bit reversing commutator

of *FF3* are high. The first incoming pulse causes *FF1* to change state, and its inverting output goes high. As a result, a positive pulse is applied to the input of *FF2* via the *OR2* gate, which causes *FF2* to change state. As a result the inverting output of *FF2* goes low, and a negative pulse appears at the input to the *AND1* gate. The duration of the pulses applied to *In1* and, at the same time, to the control input of *AND1* is chosen such that *AND1* will remain open for the time necessary for the flip-flops *FF1* and *FF2* to operate. Then, as a result of the negative pulse appearing at the input to *AND1* from the inverting output of *FF2*, the *AND1* gate will deliver a signal which causes *FF3* to change state.

This sequence of events drives the non-inverting output of *FF1* low, and the non-inverting output of *FF2* high. In this way, the flip-flops encode the binary number 100. The second incoming pulse will cause *FF1* to change state, but *FF2* and *FF3* will remain in their previous states. The new state for the flip-flops will

be 000. The third incoming pulse will cause *FF1* to change state again, and the rise of potential at its inverting output will cause *FF2* to change state so that it will reset *FF1* via *OR1*. *FF3* will remain in the previous state, because a positive pulse is generated in the feedback path, while *AND1* responds only to negative pulses. Now the states of the flip-flops represent the binary number 010. With arrival of further pulses, the flip-flops will assume states encoding the binary numbers 101, 001, 011, etc.

From the outputs of *FF1* and *FF2*, the signals go to the inputs of the *NAND* gates which also accept signals from *FF3*. The *NAND* gates perform the logic operations:

$$Out\ FF1\ (FF2) \cdot Out\ FF3 + Out\ FF1\ (FF2) \cdot FF3$$

The signals appearing on output lines *I*, *II*, and *III* are routed via amplifiers to the respective windings of the step motor.

The choice of a step motor for a number-to-shaft position converter is governed by the considerations of the desired accuracy, speed and output power.

Appendix

Information Code to GOST 10859-64

Nos.	Binary code	Character	Character symbol	Punched card code	Alphameric printer character	Typewriter, card punch and punch tape characters
1	0000000	Null	0	0	0	0
2	0000001	One	1	1	1	1
3	0000010	Two	2	2	2	2
4	0000011	Three	3	3	3	3
5	0000100	Four	4	4	4	4
6	0000101	Five	5	5	5	5
7	0000110	Six	6	6	6	6
8	0000111	Seven	7	7	7	7
9	0001000	Eight	8	8	8	8
10	0001001	Nine	9	9	9	9
11	0001010	Plus	+	0-2-8	+	+
12	0001011	Minus	−	0-3-8	−	−
13	0001100	Fractional bar	/	0-4-8	/	/
14	0001101	Comma	,	0-5-8	,	,
15	0001110	Period	.	0-6-8	.	.
16	0001111	Space		0-7-8	Space	Space
17	0010000	Decimal radix	10	12	10	10
18	0010001	Upward arrow	↑	12-0-1	↑	↑
19	0010010	Left-hand parenthesis	(	12-0-2	(	(
20	0010011	Right-hand parenthesis	)	12-0-3	)	)
21	0010100	Multiplication	×	12-0-4	×	×
22	0010101	Equals	=	12-0-5	=	=
23	0010110	Semi-colon	;	12-0-6	;	;
24	0010111	Left-hand bracket	[	12-0-7	[	[
25	0011000	Right-hand bracket	]	12-0-8	]	]
26	0011001	Asterisk	*	12-0-9	*	*
27	0011010	Left-hand quotation mark		12-2-8	‘	
28	0011011	Right-hand quotation mark	,	12-3-8	,	,
29	0011100	Not equal	≠	12-4-8	≠	≠
30	0011101	Less than	<	12-5-8	<	<
31	0011110	Greater than	>	12-6-8	>	>
32	0011111	Colon	:	12-7-8	:	:

Appendix (continued)

Nos.	Binary code	Character	Character symbol	Punched card code	Alphameric printer character	Typewriter, card punch and punch tape characters
33	0100000	Russian upper-case letters	*А*	11	*А*	*А*
34	0100001		*Б*	11-0-1	*Б*	*Б*
35	0100010		*В*	11-0-2	*В*	*В*
36	0100011		*Г*	11-0-3	*Г*	*Г*
37	0100100		*Д*	11-0-4	*Д*	*Д*
38	0100101		*Е*	11-0-5	*Е*	*Е*
39	0100110		*Ж*	11-0-6	*Ж*	*Ж*
40	0100111		*З*	11-0-7	*З*	*З*
41	0101000		*И*	11-0-8	*И*	*И*
42	0101001		*Й*	11-0-9	*Й*	*Й*
43	0101010		*К*	11-2-8	*К*	*К*
44	0101011		*Л*	11-3-8	*Л*	*Л*
45	0101100		*М*	11-4-8	*М*	*М*
46	0101101		*Н*	11-5-8	*Н*	*Н*
47	0101110		*О*	11-6-8	*О*	*О*
48	0101111		*П*	11-7-8	*П*	*П*
49	0110000		*Р*	12-11-0	*Р*	*Р*
50	0110001		*С*	12-11-1	*С*	*С*
51	0110010		*Т*	12-11-2	*Т*	*Т*
52	0110011		*У*	12-11-3	*У*	*У*
53	0110100		*Ф*	12-11-4	*Ф*	*Ф*
54	0110101		*Х*	12-11-5	*Х*	*Х*
55	0110110		*Ц*	12-11-6	*Ц*	*Ц*
56	0110111		*Ч*	12-11-7	*Ч*	*Ч*
57	0111000		*Ш*	12-11-8	*Ш*	*Ш*
58	0111001		*Щ*	12-11-9	*Щ*	*Щ*
59	0111010		*Ы*	12-11-0-2-8	*Ы*	*Ы*
60	0111011		*Ь*	12-11-0-3-8	*Ь*	*Ь*
61	0111100		*Э*	12-11-0-4-8	*Э*	*Э*
62	0111101		*Ю*	12-11-0-5-8	*Ю*	*Ю*
63	0111110		*Я*	12-11-0-6-8	*Я*	*Я*
64	1111111	Correction or emphasis		12-11-0-3-7-3-9		
65	0111111	Upper-case Roman letters	*D*	12-11-0-7-8	*D*	*D*
66	1000000		*F*	0-3-9	*F*	*F*
67	1000001		*G*	1-3-9	*G*	*G*
68	1000010		*I*	2-3-9	*I*	*I*
69	1000011		*J*	0-1-2-3-9	*J*	*J*
70	1000100		*L*	3-4-9	*L*	*L*

Appendix (continued)

Nos.	Binary code	Character	Character symbol	Punched card code	Alphameric printer character	Typewriter, card punch and punch tape characters
71	1000101		*N*	3-5-9	*N*	*N*
72	1000110		*Q*	3-6-9	*Q*	*Q*
73	1000111		*R*	3-7-9	*R*	*R*
74	1001000	Upper-case	*S*	3-8-9	*S*	*S*
75	1001001	Roman letters	*U*	0-1-3-8-9	*U*	*U*
76	1001010		*V*	0-2-3-8-9	*V*	*V*
77	1001011		*W*	1-2-3-8-9	*W*	*W*
78	1001100		*Z*	0-3-4-8-9	*Z*	*Z*
79	1001101	Overscoring	—	0-3-5-8-9	—	—
80	1001110	Less than or equal	⩽	0-3-6-8-9		⩽
81	1001111	Greater than or equal	⩾	0-3-7-8-9		⩾
82	1010000	Logic OR	∨	12-3-9		∨
83	1010001	Logic AND	∧	12-0-1-3-9		∧
84	1010010	Implication	⊐	12-0-2-3-9		⊐
85	1010011	Logic NOT	¬	12-1-2-3-9		¬
86	1010100	From ... to	÷	12-0-3-4-9		÷
87	1010101	Identity	≡	12-0-3-5-9		≡
88	1010110	Per cent	%	12-0-3-6-9		
89	1010111	Rhomb	◇	12-0-3-7-9		◇
90	1011000	Vertical bar	|	12-0-3-8-9		|
91	1011001	Horizontal bar	—	12-1-3-8-9		—
92	1011010	Underscoring	—	12-2-3-8-9		—
93	1011011	Exclamation mark	!	12-0-1-2-3-8		!
94	1011100	Quotation marks	"	12-3-4-8-9		
95	1011101		ъ	12-3-5-8-9		
96	1011110	Degrees	°	12-3-6-8-9		
97	1011111	Apostrophe	'	12-3-7-8-9		
98	1100000	Arrow to left	←	11-3-9		
99	1100001	Arrow to right	→	11-0-1-3-9		
100	1100010	Question	?	11-0-2-3-8		
101	1100011	Downward arrow	↓	11-1-2-3-9		
102	1100100	Diameter symbol	∅	11-0-3-4-9		
103	1100101	Plus or minus	±	11-0-3-5-9		
104	1100110	Surface finish symbol	▽	11-0-3-6-9		
105	1100111			11-0-3-7-9		
106	1101000			11-0-3-8-9		

Appendix (continued)

Nos.	Binary code	Character	Character symbol	Punched card code	Alphameric printer character	Typewriter, card punch and punch tape characters
107	1101001			11-2-3-8-9		
108	1101010			11-2-3-8-9		
109	1101011			11-0-1-2-3-8-9		
110	1101100			11-3-4-8-9		
111	1101101			11-3-5-8-9		
112	1101110			11-3-6-8-9		
113	1101111			11-3-7-8-9		
114	1110000			12-11-0-3-9		
115	1110001			12-11-1-3-9		
116	1110010			12-11-2-3-9		
117	1110011			12-11-0-1-2-3-9		
118	1110100			12-11-3-4-9		
119	1110101			12-11-3-5-9		
120	1110110			12-11-3-6-9		
121	1110111			12-11-3-7-9		
122	1111000			12-11-3-8-9		
123	1111001			12-11-0-1-3-8-9		
124	1111010			12-11-0-2-3-8-9	Black colour	
125	1111011			12-11-1-2-3-8-9	Red colour	
126	1111100			12-11-0-3-4-8-9	Carriage return	
127	1111101			12-11-0-3-5-8-9	Line feed	
128	1111110			12-11-0-3-6-8-9	—	

BIBLIOGRAPHY

ALEKHIN V. N. *Funktsionalnye preobrazovately napryazhenie-kod* (Voltage-to-number converters). Higher School Publishers, 1971.

ANISIMOV B. V., CHETVERIKOV V. N. *Preobrazovanie informatsii dlya ETsVM* (Data preparation for computers). Higher School Publishers, 1968.

ANISIMOV B. V., CHETVERIKOV V. N. *Osnovy teorii proektirovaniya tsifrovykh vychislitelnykh mashin* (Theory of digital computer design). "Mashinostroyenie", 1970.

BLOKH E. L., POPOV O. V., GURIN V. Ya. *Modeli istochnika oshibok v kanalakh peredachi tsifrovoi informatsii* (Error source models for digital data communication channels). "Svyaz", 1971.

BORODIN L. F. *Vvedenie v teoriyu pomekhoustoichivogo kodirovaniya* (An introduction to a theory of noise-immune coding). "Sovetskoye radio", 1968.

GALLAGHER R. J. Low-density parity-check codes.

GUROV V. S. *et al. Osnovy peredachi dannykh po provodnym kanalam svyazi* (Principles of data communication over wire communication channels). "Svyaz", 1964.

DIVNOGORTSEV V. G., KARACHENTSEVA N. Ya., YASHIN V. M. *Peredacha dannykh v setyakh vychislitelnykh tsentrov* (Data communication within computer centre networks). "Nauka i tekhnika", 1971.

KOLESNIK V. D., MIRONCHIKOV E. T. *Dekodirovanie tsiklicheskikh kodov* (Decoding of cyclic codes). "Svyaz", 1968.

KULIKOV S. V. *et al. Diskretnye preobrazovateli signalov na tranzistorakh* (Transistor digitizers). "Energiya", 1972.

MIZIN I. A., URINSON L. S., KHRAMESHIN G. K. *Peredacha informatsii v setyakh s kommutatsiei soobshchenii* (Data communication in message-switching networks). "Svyaz", 1972.

NAZAROV M. V., KUVSHINOV B. I., POPOV O. V. *Teoriya peredachi signalov* (Theory of signal transmission). "Svyaz", 1970.

NOVIKOV V. V. *et al. Osnovy telegrafii i telegrafnye stantsii* (Fundamentals of telegraphy and telegraph stations). "Svyaz", 1970.

OVCHINNIKOV V. N. *Ustroistva avtomaticheskogo obmena informatsiei* (Automatic information exchange facilities). "Energiya", 1971.

Obshcheotraslevye rukovodyashchie metodicheskie materialy po ASUP (General methodology on management information systems). State Committee for the Press, Council of Ministers, Byelorussian Soviet Republic, 1972.

PETERSON W. W. Error Correcting Codes.

SMOLOV V. B. *et al. Universalnye elektronnye preobrazovately informatsii* (General-purpose electronic data converters). "Mashinostroyenie", 1971.

Poluprovodnikovye kodiruyushchie i dekodiruyushchie preobrazovateli napryazheniya (Semiconductor voltage coders and decoders). Edited by SMOLOV V. B. and SMIRNOV N. A. "Energiya", 1967.

SAVETA N. N. *Ustroistva vvoda i vyvoda informatsii universalnykh ETsVM* (Input-output devices for general-purpose digital computers). "Mashinostroenie", 1971.

Statistika oshibok pri peredache tsifrovoi informatsii (Error statistics in digital data communication) Collection of translations, ed. by SAMOILENKO S. I. "MIR" Publishers, 1966.

SUPRUN B. A. *Pervichnye kody* (Primary codes). "Svyaz", 1970.

TEMNIKOV F. E., AFONIN V. A., DMITRIEV V. I. *Teoreticheskie osnovy informatsionnoi tekhniki* (Basic theory of information engineering). "Energiya", 1971.

Technical manual of Akkord-1200 data communication equipment.

Technical manual of EC-1020 computer.

Technical manual of БЭСМ-6 computer.

Technical manual of MINSK-32 computer.

Technical manual of RTA-6 teleprinter.

UDALOV A. P., SUPRUN B. A. *Izbytochnoye kodirovanie pri peredache informatsii dvoichnymi kodami* (Redundancy in binary data coding). "Svyaz", 1964.

USKOV N. F. *et al. Vychislitelnye klavishnye i perforiruyushchie mashiny* (Keyboard and punched-card computers). "Statistika", 1965.

FINK L. N. A Theory of Digital Message Transmission.

KHARKEVICH A. A. *Borba s pomekhami* (Noise and interference control). "Nauka", 1965.

SHASTOVA G. A. *Kodirovanie i pomekhoustoichivost peredachi telemekhanicheskoi informatsii* (Coding and noise immunity in telemetry). "Energiya", 1966.

SHANNON C. Works on information theory and cybernetics (Russian translation), 1963.

YURGENSON R. I. *Pomekhoustoichivost tsifrovykh sistem peredachi telemekhanicheskoi informatsii* (Noise-immunity of digital telemetry systems), "Energiya", 1971.

INDEX

Alphabet, Hamming, 85
Amount of information, 24
Amount of mutual information, 28

Bandwidth, effective, 14
Barker method, 263
Binit, 33
Bit, 33
Bound, Hamming, 80

Card:
  dual, 187
  mark-sensed, 187-88
  punched, 149
Card, punch, 157
  output, 231
  parallel, 157
  serial, 157
Card/type reader, 172
Channel:
  binary, 43
  communication, 41
    analog, 42
    capacity of, 54
    digital, 42
    rate of input to, 54
    rate of transmission over, 54
  multiplexor, 207, 218
  selector, 207, 220
Channel capacity, 57
Chebyshev polynomial, 21
Code, 10, 64
  Baudot, 65
  binary,
    cyclic, 266
    Gray, 268
    reflected, 268
  block, 74
  cyclic, 88
  equal-length, 64
  error-correcting, 72
  error-detecting, 72
  Hamming, 85
  information (USSR), 68, 365
  international No. 2, 68
  ISO seven-bit, 69
  Morse, 64
  nonblock, 74
  nonsystematic, 74
  nonuniform-length, 64
  primary, 64
  recurrent, 99
  simple, 64
  systematic, 74, 81
  telegraph, 64
  unequal-length, 64
  uniform-length, 64
Code mask, ternary, 272
Code redundancy, 72
  absolute, 80
  relative, 80
Code wheel:
  binary, 361
  function, 273
  ternary, 272
Coder:
  double error correcting, 100
  for cyclic codes, 92
  for recurrent codes, 99
  for systematic codes, 86
  multispeed, 292
Coding, 10
  matrix, 274
Comparator, 326
Conversion:
  analog-to-digital, 256
    error of, 257
    indirect, 277
    ramp-compare method, 320
    staircase-compare method, 324
  digital-to-analog, 338
    number-to-shaft position, 356
    serial binary number-to-voltage, 355
    step-motor technique, 361
  feedback technique, 356

Converter:
A/D, 256
code-wheel, 258
CRT, 335
feedback, 333
incremental, 258
magnetic-drum, 278
multispeed, 292
resolver phase-shift, 279
selective-subtraction, 329
total-value, 260
voltage-to-time-to-digital, 320
D/A, 255
current-summation, 349
two-resistance-value, 351
using d. c. amplifier, 345
weighted-resistor, 340
weighted-voltage, 338

Data:
entry from punched cards, 221
preparation for entry to computer, 148
Data communication equipment, 138
Data logger, 198
Data transfer:
first-generation computers, 205
second-generation computers, 206
third-generation computers, 209
Decoder:
for cyclic codes, 92
for recurrent codes, 101
for systematic codes, 87
majority, 96
probabilistic, 116
Decoding:
erasure, 110
majority, 94
probabilistic, 112
redundant digital signals, 103
reliable symbols, 109
sequence estimation, 103
symbol estimation
Decoding tree, 114
Demodulation, 43, 51
Delta modulation, 24
Detection, Wagner's scheme, 109
Distance, Hamming, 79
Distortion:
irregular, 53
regular, 53
Dual-brush method, 263

Electrophotography, 242
Entropy:
continuous case, 31
differential, 33
ε-, 34
relative, 33
source, 25
unit of, 33
Error detection, probabilistic-algebraic, 117

Flash tube, 309
Function, bandlimited, 13

Grating, optical, 313

Inductosyn, 282
Information:
amount of, 24
definition, 9
mutual, 29
Input preparation:
automatic, 187
equipment for, 182
punched-card, 182
punched-tape, 184
Interference, 53
Interrupt:
priority of, 214
hardware controlled, 214
software controlled, 214
timing of, 213

Kotelnikov's sampling theorem, 13

Light detector, 301
Light source, 307
Line, communication, 41
Link, communication, 41
Lipschitz'condition, 36
Loop:
local, 43
subscriber's, 43

Machine document preparation, 148
Magnetography, 243
Markov process, 29

Matrix:
generator, 82
parity-check, 82
Medium, communication, 41
Message, 9, 41
Modem, 139
Modulation, 43
amplitude, 44
angle, 46
frequency, 45
phase, 47
pulse, 48
pulse-duration, 49
pulse-frequency, 51
pulse-position, 50
pulse-width, 49
Modulation factor, 44
Multiplexor, terminal, 249
Multiplexor channel, 218
Multiprogramming, 216

Network, communication, 58
Noise, 53
additive, 53
continuous, 54
fluctuation, 54
impulse, 54
random, 54
multiplicative, 53
white, 54

Parity check, 73
equation, 85
rule, 85
sequence, 84
Perforator, 166
Photodiode, 301
Photoresistor, 306
Phototransistor, 301
Photovoltaic cell, 301
Printout, alphanumeric, 240
Printing:
electrolytic, 241
electrostatic, 242
Printing mechanism:
type-bar, 179, 180
type-wheel, 179, 180
on-the-fly, 180
Processing mode:
conversational, 216
interactive, 216
remote batch, 216
Program interrupt, 212
Pulse-code modulation (PCM), 23

Quantization:
amplitude, 12, 22
time, 12
Quantizing, 12, 22

Ramp generator, 322
Rate of input, 54
Rate of transmission, 54
Reader:
air-jet, 173
mechanical, 173
Reading:
brush, 297
electric-contact, 297
electrostatic, 173
induction, 311
methods of, 297
photoelectric, 299
pneumatic, 173
Recording:
dry electrosensitive, 241
Resolver:
capacitive, 287
induction, 279
Rolle's theorem, 19

Sampling, 12, 18, 23
rate of, 16
Scheduler, 217
Selector channel, 220
Shannon, 25
encoding theorem, 57
sampling theorem, 13
Signal:
analog, 12
continuous, 11
definition, 9
discrete, 11
dynamic, 11
static, 11
Staircase generator, 325
Source rate, 31
Supervisor, 217
Synchro, 281

System, communication, 40
adaptive-feedback, 134
address-RQ, 120, 132
combination-RQ, 127
data-feedback, 120, 121
decision-feedback, 120, 125
feedback, 119
receiver-interlock, 128
symbol-RQ, 125
time-sharing, 216
two-way, 119

Tape, punched, 153
Tape punch, 166
output, 231
Teleprinter, 192

Flash tube, 309
Typewriter, electric, 175
Unit, binary, 33

V-brush method, 263

TO THE READER

Mir Publishers welcome your comments on the content, translation, and design of the book.

We would also be pleased to receive any suggestions you care to make about our future publications.

Our address is:

USSR, 129820, Moscow, I-110, GSP, Pervy Rizhsky Pereulok, 2, Mir Publishers

Printed in the Union of Soviet Socialist Republics

OTHER BOOKS FOR YOUR LIBRARY

CYBERNETIC METHODS IN CHEMISTRY AND CHEMICAL ENGINEERING

BY V. KAFAROV

In a simple and fairly popular manner, the book presents the basic concepts of cybernetics, and describes its methods and tools (computers) as applied to chemistry and chemical engineering. Separate chapters are devoted to the principles of a cybernetic approach to the analysis of chemical-engineering processes and the development of new processes. A detailed consideration is given to the use of mathematical models for processes and typical reactors used in chemical engineering on the basis of reaction kinetics and heat transfer. Special mention is made of scale-up procedures and examples are given of optimum designs.

AN INTRODUCTION TO COMPUTERS

BY N. SERGEEV, N. VASHKEVICH

This book covers the circuitry and operating principles of analog and digital computers, special-purpose computing devices, machines and systems. Ample space is devoted to a basic theory of similitude and simulation, and a fairly detailed description is given of basic functional elements, assemblies and units which make up typical analog and digital computers, in conjunction with an introduction to the mathematical and logic basis of electronic digital computers.

Separate sections deal with block-diagram synthesis of analog computers to solve algebraic, transcendental, ordinary and partial differential equations and their systems.

Prof. Vladimir CHETVERIKOV, D. Sc. (Tech.), has been Head of the Department for Automation and Control Systems at the Baumann Higher Technical School (Moscow). Graduated from the same school in 1952. He has written many textbooks and monographs on computation, management information and control systems. Now he heads the Institute of Fundamental Research for the Higher School.